Programmation bayésienne des robots

Olivier Lebeltel, Pierre Bessiere, Julien Diard, Emmanuel Mazer

Programmation bayésienne des robots

Olivier Lebeltel — Pierre Bessière — Julien Diard — Emmanuel Mazer

Laboratoire GRAVIR / IMAG
INRIA Rhône-Alpes
ZIRST – 655 Av. de l'Europe
38330 Montbonnot St Martin

Olivier.Lebeltel@inrialpes.fr

RÉSUMÉ. *Cet article propose une méthode originale de programmation des robots fondée sur l'inférence et l'apprentissage bayésien. Cette méthode traite formellement des problèmes d'incertitude et d'incomplétude inhérents au domaine considéré. La principale difficulté de la programmation des robots vient de l'inévitable incomplétude des modèles utilisés. Nous exposons le formalisme de description d'une tâche robotique ainsi que les méthodes de résolution. Nous l'illustrons en utilisant ce système pour programmer une application de surveillance pour un robot mobile : le Khepera. Pour cela, nous utilisons des ressources génériques de programmation appelées « descriptions ». Nous montrons comment définir et utiliser de manière incrémentale ces ressources (comportements réactifs, fusion capteur, reconnaissance de situations et séquences de comportements) dans un cadre systématique et unifié.*

ABSTRACT. *This paper proposes an original method for robotic programming based on bayesian inference and learning. This method formally deals with problems of uncertainty and incomplete information that are inherent to the field. Indeed, the principal difficulties of robot programming comes from the unavoidable incompleteness of the models used. We present the formalism for describing a robotic task as well as the resolution methods. We illustrate it by programming a surveillance task with a mobile robot: the Khepera. In order to do this, we use generic programming resources called "descriptions". We show how to define and use these resources in an incremental way (reactive behaviors, sensor fusion, situation recognition and sequences of behaviors) within a systematic and unified framework.*

MOTS-CLÉS : *robotique autonome, inférence bayésienne, programmation des robots.*

KEYWORDS: *autonomous robotic, robot programming, bayesian inference.*

1. Introduction

Nous prenons pour postulat que toute modélisation d'un phénomène réel est irrémédiablement *incomplète*. Il existe toujours des variables cachées, non prises en compte dans le modèle, qui influencent le phénomène. L'effet de ces variables cachées est que le modèle et le phénomène n'ont jamais le même comportement. Tout système robotique est confronté à cette difficulté centrale : comment utiliser un modèle incomplet par rapport à son environnement pour percevoir, inférer, décider et agir efficacement ? Nous proposons une méthode de programmation de robots originale qui répond spécifiquement à cette question. Le raisonnement rationnel à partir d'informations incomplètes est un défi pour les systèmes artificiels. Le but de l'inférence et de l'apprentissage bayésien est précisément d'aborder ce problème avec une théorie formelle et bien établie. La Programmation Bayésienne des Robots (PBR) repose sur ce cadre bayésien. Nous présentons quelques exemples de programmation qui illustrent cette approche et définissent les descriptions comme ressources de programmation génériques. Nous montrons que ces ressources peuvent être utilisées pour construire de manière incrémentale des systèmes complexes dans un cadre systématique et uniforme. Le système repose sur la base simple et fiable de l'inférence bayésienne. Il oblige le programmeur à déclarer explicitement toutes les hypothèses qui ont été faites. Finalement, il permet un traitement efficace de l'information incomplète et incertaine dans l'élaboration de programmes robotiques.

Cet article est organisé de la façon suivante : la section 2 offre un court passage en revue des travaux connexes, la section 3 est dédiée aux définitions et aux notations. La section 4 présente la plate-forme expérimentale puis différents exemples de programmes bayésiens : apprentissage de simples comportements réactifs, combinaisons de comportements, fusion de capteurs, reconnaissance de situation et séquences temporelles. Tous ces exemples forment les ressources qui seront finalement combinées pour programmer un robot afin d'accomplir une tâche de surveillance de nuit. Enfin, section 5, nous concluons avec une synthèse qui résume les principes, les fondements théoriques et la méthode de programmation.

2. État de l'art

Nos travaux sont basés sur une implémentation du principe de la théorie bayésienne des probabilités.

En physique, depuis les travaux précurseurs de Laplace (Laplace, 1774 ; Laplace, 1814), de nombreux résultats ont été obtenus en utilisant les techniques d'inférence bayésienne (afin de prendre en compte l'incertitude) et le principe de maximum d'entropie (afin de prendre en compte l'incomplétude). Edward T. Jaynes a proposé une formalisation synthétique et rigoureuse du raisonnement probabiliste avec sa théorie « Probability as Logic » (Jaynes, 2003). Un bilan historique de cette

approche a été donné par Jaynes (Jaynes, 1979) et une analyse épistémologique par Matalon (Matalon, 1967). Les justifications théoriques de l'inférence probabiliste et du maximum d'entropie sont nombreuses. Les théorèmes de concentration d'entropie (Jaynes, 1982 ; Robert, 1990) sont parmi les plus rigoureux, le théorème de Cox (Cox, 1961) étant le plus connu, bien qu'il ait été partiellement discuté par Halpern (Halpern, 1999a ; Halpern, 1999b). De nombreux outils mathématiques et applications ont été développés (Smith et al., 1985 ; Tarentola, 1987 ; Bretthorst, 1988 ; Erickson et al., 1988a,1988b ; Mohammad-Djafari et al., 1992 ; Kapur et al., 1992).

En intelligence artificielle, l'importance du raisonnement avec une connaissance incertaine a été reconnue depuis longtemps. Cependant, l'approche bayésienne ait apparut clairement comme un principe « à la mode » seulement depuis la proposition de réseaux Bayésiens (Pearl, 1988) et des modèles graphiques (Lauritzen et al., 1988 ; Lauritzen, 1996 ; Jordan, 1998 ; Frey, 1998). Il a été prouvé que l'inférence Bayésienne est un problème NP difficile (Cooper, 1990). Cependant, des progrès techniques faits récemment permettent des calculs approximatifs en un temps raisonnable (Saul et al., 1996 ; Zhang et al., 1996 ; Delcher et al., 1996 ; Darwiche et al., 1997 ; Koller et al., 1997 ; Ruiz et al., 1998 ; Jaakkola et al., 1999 ; Jordan et al., 1999).

En général, les architectures récentes de programmation de robots (Alami et al., 1998 ; Borrelly et al., 1998 ; Schneider et al., 1998; Dekhil et al., 1998 ; Mazer et al., 1998) n'abordent pas le problème de l'incertitude. En robotique, l'incertitude est soit rattachée à la calibration (Bernhardt et al., 1993) soit aux problèmes de planification (Brafman et al., 1997). Dans ce dernier cas, des auteurs ont pensé à élaborer un modèle de l'incertitude des mouvements du robot dans des problèmes de planification d'opérations d'assemblage (Lozano-Perez et al., 1984 ; Donald, 1988) ou représenter l'incertitude liée à la position du robot dans un lieu (Kapur et al., 1992). Plus récemment, les techniques bayésiennes ont été largement utilisées dans le cadre des PDMPO (Processus de Décision Markoviens Partiellement Observables) pour planifier des chemins complexes dans des environnements partiellement connus (Kaelbling et al., 1996a, 1996b, 1998 ; Lane et al., 2001) ou pour la sélection d'actions (Rosenblatt, 2000). Les modèles de Markov cachés sont aussi utilisés pour planifier des tâches et reconnaître des situations dans des environnements complexes (Aycard, 1998 ; Thrun, 1998). Enfin, de nombreux travaux ont été faits sur la localisation probabiliste, à base de grilles d'occupation probabiliste (Konolidge, 1997), avec des méthodes de Markov (Thrun et al., 1998 ; Gutmann, et al., 1998 ; Fox et al., 2000) ou encore de la localisation markovienne à base de corrélation (Konolidge et al., 1999). Cependant, à la lumière des connaissances actuelles, la conception d'un système et d'une architecture de programmation pour robots basés uniquement sur l'inférence bayésienne n'a jamais été étudiée avant la thèse de Lebeltel (Lebeltel, 1999 ; Diard & Lebeltel, 1999 ; Lebeltel et al., 2000 ; Diard & Lebeltel, 2000). Un article de Thrun (Thrun, 2000) explore une direction similaire avec toutefois moins de généralité. PBR propose un

cadre simple et générique pour la programmation des robots en présence d'incertitude et d'incomplétude. Au-delà d'applications robotiques, le formalisme de PBR autorise une reformulation et une comparaison de nombreux modèles probabilistes standards comme les Réseaux Bayésiens, les Réseaux Bayésiens Dynamiques, les Filtres Bayésiens, les Modèles de Markov Cachés, Filtre de Kalman. (voir (Bessière et al., 2003) pour une étude détaillée). Enfin, une présentation des fondements épistémologiques de la PBR peut être trouvée dans deux articles (Bessière et al., 1998a, 1998b) et tous les détails techniques dans (Lebeltel, 1999).

3. Concepts de base

3.1. *Définitions et notations*

3.1.1. Propositions et Variables

Le premier concept que nous utiliserons est la notion usuelle de *proposition logique*. Les propositions seront notées par une minuscule italique. Elles peuvent être composées en utilisant les opérateurs logiques usuels : $a \wedge b$ dénote la conjonction des propositions a et b, $a \vee b$ leur disjonction et $\neg a$ la négation de la proposition a. La notion de variable discrète est le second concept que nous utiliserons. Les variables seront dénotées par des noms débutant par une lettre majuscule. Une variable discrète $\mathtt{V}$ est associée à un domaine de valeurs $D_{\mathtt{V}} = \{\mathtt{v}_1, \ldots, \mathtt{v}_k\}$, l'ensemble des k valeurs que peut prendre cette variable. $\lfloor \mathtt{V} \rfloor$ dénote le cardinal de l'ensemble des valeurs possibles pour une variable discrète $\mathtt{V}$ ($\lfloor \mathtt{V} \rfloor = k$). Par définition, une variable discrète $\mathtt{V}$ est un ensemble de k propositions logiques $v_i. : v_i \equiv [\mathtt{V} = \mathtt{v}_i]$ (« la variable $\mathtt{V}$ a pour valeur $\mathtt{v}_i$ »). Ces propositions sont mutuellement exclusives (pour tout i,j tels que $i \neq j$, $v_i \wedge v_j$ est faux*)* et exhaustive (au moins une des propositions v_i est vraie). La conjonction de deux variables $\mathtt{X}$ et $\mathtt{Y}$, notée $\mathtt{X} \otimes \mathtt{Y}$, définit un ensemble de $\lfloor \mathtt{X} \rfloor \times \lfloor \mathtt{Y} \rfloor$ propositions $x_i \wedge y_j$. $\mathtt{X} \otimes \mathtt{Y}$ dénote un ensemble de propositions logiques exhaustives et mutuellement exclusives et à ce titre peut considérée comme une nouvelle variable[1]. Par extension, la conjonction de n variables est également une variable, et peut donc être renommée et considérée par la suite comme une unique variable.

3.1.2. Probabilités

La notion de probabilité intervient lorsque nous ne sommes pas en mesure de connaître de manière certaine la valeur de vérité d'une proposition. Nous définissons intuitivement la probabilité comme le degré de certitude que nous avons dans la

1. En revanche, la disjonction de deux variables, définie comme l'ensemble des propositions $V_i \vee V_j.$, ne dénote pas une variable ; les propositions associées ne sont pas mutuellement exclusives.

véracité d'une proposition. Ce degré de certitude est fortement lié à un ensemble de connaissances contextuelles dont nous disposons et sur la base desquelles nous souhaitons juger de sa véracité. Nous notons cet ensemble de connaissances par ω. De manière générale, ces connaissances seront le fruit d'un ensemble de *connaissances préalables* (noté π) et d'un ensemble de paramètres (noté λ) dont les valeurs sont soit identifiées à partir d'un ensemble de *données expérimentales* (noté δ), soit fixées a priori (noté λ_0). Ainsi, nous préciserons toujours les connaissances qui nous permettent d'assigner une probabilité à une proposition *A*. Par conséquent, la probabilité d'une proposition *A* sera toujours au moins conditionnée par ω : $\mathbf{P}(A \mid \omega)$. Pour chaque ensemble de connaissances ω, $\mathbf{P}(\cdot \mid \omega)$ est une application qui assigne à chaque proposition *A* une unique valeur réelle $\mathbf{P}(A \mid \omega)$ dans l'intervalle [0, 1].

Nous cherchons à raisonner sur les probabilités des conjonctions, disjonctions et négations de propositions, dénotées respectivement par $\mathbf{P}(a \wedge b \mid \omega)$, $\mathbf{P}(a \vee b \mid \omega)$ et $\mathbf{P}(\neg a \mid \omega)$. Nous nous intéresserons également à la probabilité d'une proposition *A* conditionnée par la connaissance ω et la connaissance de la valeur de vérité d'une autre proposition *b* ; ceci sera dénoté par $\mathbf{P}(a \mid b \wedge \omega)$ (« la probabilité conditionnelle que *a* est vraie, sachant que *b* est vraie dans le contexte ω »). Dans l'intérêt d'éviter des problèmes impossibles, nous ne nous permettrons pas de raisonner à partir de prémisses impossibles ou mutuellement contradictoires. Par conséquent, nous ne chercherons pas à définir $\mathbf{P}(a \mid b \wedge c \wedge \omega)$ lorsque que *b* et *c* sont des propositions mutuellement contradictoires. Quand de telles notations se présenteront, il sera implicitement compris que les propositions *b* et *c* sont compatibles.

Pour des raisons de simplicité et de clarté, nous écrirons des formules de probabilités portant sur des variables au lieu de propositions. Par convention, chaque fois qu'une variable $\mathtt{V}$ apparaît dans une formule de probabilité $\Phi(\mathtt{V})$, cette formule doit être interprétée comme un raccourci d'écriture pour $\forall\ \mathtt{v}_i \in D_{\mathtt{V}}$, $\Phi(\mathtt{v}_i)$. Par exemple, l'expression $\mathbf{P}(\mathtt{X} \otimes \mathtt{Y} \mid \mathtt{Z} \otimes \omega) = \mathbf{P}(\mathtt{X} \mid \omega)$ dénote, étant donné trois variables $\mathtt{X}$, $\mathtt{Y}$ et $\mathtt{Z}$:

$$\forall \mathtt{x}_i \in D_{\mathtt{X}}, \forall \mathtt{y}_j \in D_{\mathtt{Y}}, \forall \mathtt{z}_k \in D_{\mathtt{Z}},$$
$$\mathrm{P}\left(x_i \wedge y_j \mid z_k \wedge \omega\right) = \mathrm{P}\left(x_i \mid \omega\right)$$

3.2. *Postulat et règles d'inférences*

Le raisonnement probabiliste repose sur deux règles de base :

– La *règle du produit* donne la probabilité de la conjonction de deux propositions *A* et *B* :

$$\mathrm{P}\left(a \wedge b \mid \omega\right) = \mathrm{P}\left(a \mid \omega\right)\mathrm{P}\left(b \mid a \wedge \omega\right) = \mathrm{P}\left(b \mid \omega\right)\mathrm{P}\left(a \mid b \wedge \omega\right) \quad [1]$$

– La *règle de normalisation* exprime la relation entre les probabilités d'une proposition A et de sa négation $\neg A$:

$$\mathrm{P}\left(a \mid \omega\right) + \mathrm{P}\left(\neg a \mid \omega\right) = 1 \quad [2]$$

Dans le cadre de cet article, nous considérons la *règle du produit* [1] et la *règle de normalisation* [2] comme des postulats. Ces règles forment la base à partir de laquelle tous les calculs de probabilités dérivent. En effet, partant de ces deux règles [1] et [2], nous pouvons dériver les trois règles d'inférence suivantes concernant les variables :

– 1. règle du produit pour les variables

$$\mathrm{P}\left(\mathtt{X} \otimes \mathtt{Y} \mid \omega\right) = \mathrm{P}\left(\mathtt{X} \mid \omega\right) \times \mathrm{P}\left(\mathtt{Y} \mid \mathtt{X} \otimes \omega\right) = \mathrm{P}\left(\mathtt{Y} \mid \omega\right) \times \mathrm{P}\left(\mathtt{X} \mid \mathtt{Y} \otimes \omega\right) \quad [3]$$

– 2. règle de la normalisation pour les variables

$$\sum_{\mathtt{X}} \mathrm{P}\left(\mathtt{X} \mid \omega\right) = 1 \quad [4]$$

– 3. règle de marginalisation pour les variables

$$\sum_{\mathtt{X}} \mathrm{P}\left(\mathtt{X} \otimes \mathtt{Y} \mid \omega\right) = \mathrm{P}\left(\mathtt{Y} \mid \omega\right) \quad [5]$$

3.3. *Programmes bayésiens*

Nous définissons un programme bayésien comme un moyen de spécifier une famille de distributions de probabilités. Notre objectif est de montrer qu'il est possible, en utilisant de telles spécifications, de définir des tâches robotiques, de réaliser de l'apprentissage et enfin de commander effectivement un robot.

La Figure 1 présente les éléments qui constituent un programme bayésien :

– Un *programme* est construit à partir d'une *description* et d'une *question*.

– Une *description* est définie par des *connaissances préalables* et les valeurs des *paramètres* libres.

– Les *connaissances préalables* décrivent un ensemble de *variables* pertinentes, une *décomposition* et un ensemble de *formes*.

– Les *formes* sont soit *paramétriques* soit des *questions* posées à une autre description.

– Les valeurs des paramètres sont soit fixées a priori soit obtenues par un mécanisme d'identification à partir d'un ensemble de *données expérimentales*.

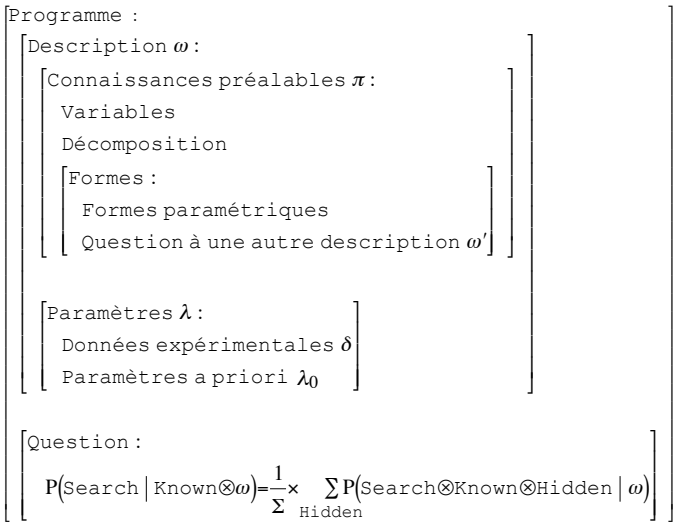

Figure 1. *Structure d'un programme bayésien*

3.3.1. Description

L'objet d'une description ω est de spécifier une méthode effective pour calculer la distribution conjointe d'un ensemble de variables {V1, V2, …, Vn}. Cette distribution conjointe est dénotée par : **P**(V1 ⊗ V2 ⊗ … ⊗ Vn | ω).

3.3.2. Connaissances préalables

Afin de définir les connaissances préalables, le programmeur doit spécifier les points suivants :

– 1. Déterminer l'ensemble des *variables* pertinentes {V1, V2, …, Vn} sur lesquelles sera définie la distribution conjointe.

– 2. Décomposer la distribution conjointe.

Étant donné une partition de l'ensemble des variables {V1, V2, …, Vn} en *k* sous-ensembles, nous pouvons définir *k* variables L1, L2, …, Lk chacune d'elle correspondant à un de ces sous-ensembles. Chacune des variables Li est la conjonction des variables Vi1, Vi2, … qui sont éléments du sous-ensemble *i*. La

règle du produit [4] permet de décomposer la distribution conjointe de la manière suivante :

$$P(V1 \otimes \ldots \otimes Vn \mid \omega) = P(L1 \mid \omega) \times P(L2 \mid L1 \otimes \omega) \times \ldots \times P(Lk \mid Lk-1 \otimes \ldots \otimes L2 \otimes L1 \otimes \omega)$$

Enfin, l'expression des hypothèses d'indépendance conditionnelle permet de simplifier d'autant plus la décomposition. Une hypothèse d'indépendance conditionnelle pour une variable Li permet de sélectionner certaines variables Vj parmi les variables comprises dans la partie conditionnelle, Li–1 ⊗ … ⊗ L2 ⊗ L1 ; soit Ri la conjonction de ces variables choisies. Une hypothèse d'indépendance conditionnelle s'exprime de la manière suivante :

$$P(Li \mid Li-1 \otimes \ldots \otimes L2 \otimes L1 \otimes \omega) = P(Li \mid Ri \otimes \omega)$$

Nous obtenons ainsi :

$$P(V1 \otimes \ldots \otimes Vn \mid \omega) = P(L1 \mid \omega) \times P(L2 \mid R2 \otimes \omega) \times \ldots \times P(Lk \mid Rk \otimes \omega)$$

Cette expression de la distribution conjointe en un produit de distributions plus simples est appelée une décomposition.

– 3. Définir les formes.

À chaque distribution élémentaire **P**(Li | Ri ⊗ ω) doit être associée soit une forme paramétrique (i.e., une fonction f_λ(Li)), soit une question posée à une autre description. En général, λ est un vecteur de paramètres qui dépend de Ri. On parle d'apprentissage lorsque certains de ces paramètres sont calculés en tenant compte de l'ensemble de données expérimentales δ. Toutefois, il est également possible de fixer les valeurs, λ_0, des paramètres de manière a priori.

3.3.3. Question

Étant donné une description, **P**(V1 ⊗ V2 ⊗ … ⊗ Vn | ω), poser une question consiste à définir une partition de {V1, V2, …, Vn} en trois ensembles : Eq l'ensemble des variables dont on cherche la distribution de probabilité, Ec les variables dont les valeurs sont connues, et Ei les variables ignorées. Nous définissons les variables Q, c et I comme la conjonction des variables de chacun de ces ensembles. Les variables dont les valeurs sont connues seront notées par une minuscule afin de rappeler que leurs valeurs sont déterminées. Une question probabiliste consiste à évaluer l'expression suivante : **P**(Q | c ⊗ ω)

3.4. Exécution d'un programme bayésien

L'exécution d'un programme nécessite d'être en mesure de réaliser une inférence bayésienne puis de décider des valeurs particulières des variables recherchées en tenant compte de la distribution de probabilités obtenue au terme de l'inférence.

3.4.1. Inférence bayésienne

Étant donné une distribution conjointe $\mathbf{P}(\mathtt{V1} \otimes \mathtt{V2} \otimes \ldots \otimes \mathtt{Vn} \mid \omega)$, il est toujours possible de calculer n'importe quelle question en appliquant le schéma générique d'inférence suivant :

$$\begin{aligned}
P(\mathtt{Q} \mid \mathtt{c} \otimes \omega) &= \frac{P(\mathtt{Q} \otimes \mathtt{c} \mid \omega)}{P(\mathtt{c} \mid \omega)} \\
&= \frac{\sum_{\mathtt{I}} P(\mathtt{Q} \otimes \mathtt{c} \otimes \mathtt{I} \mid \omega)}{\sum_{\mathtt{I},\mathtt{Q}} P(\mathtt{Q} \otimes \mathtt{c} \otimes \mathtt{I} \mid \omega)} \\
&= \frac{1}{\Sigma_{\mathtt{c}}} \times \sum_{\mathtt{I}} P(\mathtt{Q} \otimes \mathtt{c} \otimes \mathtt{I} \mid \omega) \\
&= \frac{1}{\Sigma_{\mathtt{c}}} \times \sum_{\mathtt{I}} \left[P(\mathtt{L1} \mid \omega) \prod_{\mathtt{i}=2}^{\mathtt{k}} P(\mathtt{Li} \mid \mathtt{Ri} \otimes \omega) \right]
\end{aligned}$$

où la première égalité résulte de l'application de la règle du produit [3], la seconde et la troisième résultent de l'application de la règle de marginalisation [5]. Le dénominateur s'avère n'être qu'un terme de normalisation fonction de la valeur des variables connues, par conséquent nous adopterons comme convention de le remplacer par Σ_c. Finalement, la distribution conjointe est remplacée par sa décomposition spécifiée dans les connaissances préalables. Deux problèmes doivent être résolus : trouver les pics de probabilité dans l'espace des variables recherchées et marginaliser dans l'espace des variables ignorées. Le calcul exhaustif ne peut être envisagé que pour un espace de recherche de dimension réduite. Dès lors que le nombre de cas devient prohibitif, il est nécessaire soit de construire une représentation qui approxime la distribution recherchée, soit de réaliser des tirages directement à partir de son expression. Le problème est d'autant plus difficile que l'évaluation de l'expression pour un cas particulier des variables recherchées nécessite de marginaliser la distribution conjointe sur l'espace des variables ignorées. Par conséquent, il est nécessaire également d'approximer cette marginalisation, ce qui se ramène de nouveau à chercher les pics de probabilité sur l'espace des variables ignorées. Il est bien connu que l'inférence bayésienne générale est un problème très difficile. L'inférence exacte est NP-difficile (Cooper, 1990) et le problème général d'inférence approximée l'est également

(Dagum et al., 1993). Toutefois, il est possible en pratique de traiter cette complexité pour trois raisons principales :

– Les hypothèses d'indépendances conditionnelles telles qu'elles peuvent être exprimées dans la décomposition de la distribution conjointe diminuent la complexité du problème en réduisant de manière drastique l'espace de recherche. L'importance de la décomposition à été souligné par de nombreux auteurs (voir Zang et al., 1996).

– Des simplifications symboliques peuvent être réalisées avant la phase de calcul numérique.

– De nombreuses méthodes d'optimisation numérique et de marginalisation existent et peuvent être mises en œuvre dans le contexte de l'inférence bayésienne.

3.4.2. Décision

Pour une question donnée, différentes politiques de décision sont possibles comme, par exemple, de chercher la meilleure valeur (de plus haute probabilité) combinée avec des fonctions d'utilité, ou encore de tirer aléatoirement selon la distribution de probabilité. Par la suite, dans les différentes expériences présentées, nous appliquerons cette dernière méthode à laquelle nous ferons référence par la notation suivante : **Draw**(**P**(Q | c ⊗ ω)).

3.4.3. PL : une API pour automatiser l'inférence bayésienne

Un moteur d'inférence et l'API[2] associée (nommée PL pour Probabilistic Library) ont été développés et utilisés dans les expériences présentées dans cet article. Dans PL, l'inférence est réalisée en deux phases : une phase de simplification symbolique et une phase de calcul numérique intensif qui calcule une approximation des distributions. Le principal objectif de la simplification est de réduire la complexité des marginalisations en limitant le nombre de sommes nécessaires au calcul de la distribution :

$$P(Q \mid c \otimes \omega) = \frac{1}{\sum_c} \times \sum_I \left[P(L1 \mid \omega) \prod_{i=2}^{k} P(Li \mid Ri \otimes \omega) \right]$$

Ce type de simplification est largement appliqué dans la littérature. Dans le cadre des réseaux bayésiens, l'algorithme d'arbre de jonction (ou JLO, Jensen et al., 1990) peut être vu comme une technique particulière de simplification. Dans PL, différentes simplifications sont appliquées. Tout d'abord, si nous considérons différents termes du produit sommé, une première phase de simplification évidente peut être appliquée :

2. Application Programming Interface.

– Les termes où toutes les variables sont connues ainsi que les termes associés à une distribution uniforme sont constants. Ils disparaissent de l'expression, leur valeur sera prise en compte implicitement dans le terme de normalisation.

– Enfin, les termes constitués de variables connues ou recherchées peuvent être factorisés en dehors de la somme.

Suite à cette première étape de simplification, nous obtenons une nouvelle expression de la forme :

$$P(\mathtt{Q} \mid \mathtt{c} \otimes \omega) = \frac{1}{\sum_{\mathtt{c}}} \times \prod_{j \in J} P(\mathtt{Lj} \mid \mathtt{Rj} \otimes \omega) \times \sum_{\mathtt{I}} \left[\prod_{i \in I} P(\mathtt{Li} \mid \mathtt{Ri} \otimes \omega) \right]$$

Une deuxième étape consiste à parcourir les termes de la marginalisation afin d'éliminer ceux qui se somment à 1. En effet, les termes $\mathbf{P}(\mathtt{Li} \mid \mathtt{Ri} \otimes \omega)$ dont toutes les variables qui apparaissent dans `Li` appartiennent à Ei (variables ignorées) et dont toutes les variables qui apparaissent dans `Ri` sont connues ou recherchées, se somment à 1. Enfin, la dernière simplification est obtenue en réordonnant l'ordre des sommes sur les différentes variables ignorées afin de minimiser le nombre d'opérations. Pour cela, PL implémente l'algorithme de la loi distributive généralisée proposé par Aji et McEliece (Aji et al, 2000).

L'objectif de la phase de calcul numérique est d'estimer la distribution $\mathbf{P}(\mathtt{Q} \mid \mathtt{c} \otimes \omega)$ et, par conséquent, d'estimer la somme correspondante. Deux principales approches sont envisageables : soit en construisant une approximation explicite de ces distributions, soit en réalisant des tirages. PL comprend différents algorithmes associés à chacune de ces approches. Il est possible d'approximer la distribution par des filtres à particules (Arulampalam et al., 2002) ou par des arbres binaires multi résolution (MRBT[3]). PL propose également des techniques de tirage basées principalement sur des méthodes de Monte-Carlo (Neal, 1993 ; MacKay, 1996) et une amélioration de ces méthodes proposée par Mekhnacha (Mekhnacha et al, 2001) combinant le recuit simulé. Plus de détails sur les simplifications symboliques et les méthodes de calcul numérique mises en œuvre dans PL peuvent être trouvés dans (Bessiere et al. 2003).

4. Expérience

L'objectif de l'expérience présentée est de montrer comment l'expression d'une tâche complexe de robotique peut être obtenue sous la forme d'une description. Cette description est le résultat d'une structuration de descriptions plus élémentaires utilisées comme briques de base pour programmer.

3. Multi Resolution Binary Trees (brevet en cours, Bessière, 2002).

4.1 Plate-forme expérimentale

4.1.1. Le robot Khepera

Cette expérience a été réalisée avec un mini robot mobile : le Khepera (voir Figure 1). Ce robot à été conçu à l'E.P.F.L. et est commercialisé par la société K–team. Le Khepera est équipé de huit capteurs de luminosité (six à l'avant et deux derrière) qui prennent des valeurs comprises entre 0 et 511 inversement proportionnelles à la quantité de luminosité mesurée, enregistrées dans les variables `Lm1`, …,`Lm8`. Ces huit capteurs peuvent également être utilisés comme des capteurs de proximité infrarouge, prenant des valeurs entre 0 et 1023 inversement proportionnelles à la distance aux obstacles. Les variables associées aux capteurs de proximités sont `Px1`, …,`Px8`. Le robot est contrôlé par la vitesse de sa roue droite (variable `Md`) et de sa roue gauche (variable `Mg`).

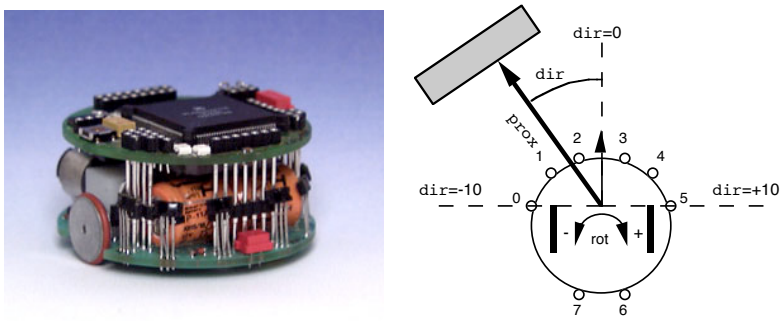

Figure 1. *Le robot mobile Khepera vu de face ; à droite, un schéma qui représente ses variables sensorielles et motrices.*

4.1.2. L'environnement

Le Khepera est placé dans une arène de 1 m par 1 m. Cet environnement est ceint de murs texturés afin d'être mieux perçu par la caméra linéaire. À l'intérieur de l'environnement, nous plaçons des murs réalisés à base de briques Lego© qui peuvent être aisément déplacés afin d'obtenir rapidement diverses configurations. Nous matérialisons la « base » du Khepera dans un des coins de l'arène par un renfoncement fait de murs plus hauts et plaçons une petite lampe au-dessus.

4.1.3. La tâche à réaliser

L'objectif de cette expérience est de programmer le Khepera afin de lui faire réaliser une tâche de surveillance. Au cours de la patrouille, le robot aura pour tâches :

– Patrouiller aléatoirement son environnement en évitant les collisions avec les obstacles.

– Revenir à sa base quand on lui en donne l'ordre.

– Gérer son énergie en rentrant régulièrement à la base pour se recharger.

– Donner l'alarme et intervenir s'il détecte un « incendie ».

La réalisation de cette tâche va être obtenue de manière modulaire. Dans un premier temps, nous décrirons les ressources comportementales de bas niveau (comportements d'évitement d'obstacle, de phototaxie, de retour à la base). La base du robot est surmontée d'une forte source lumineuse qui permet d'obtenir un comportement de retour à la base en combinant l'évitement d'obstacle et un comportement de phototaxie positive[4]. Ensuite nous décrirons deux modèles capteurs : le premier afin de déterminer la direction de la source lumineuse (variable `Theta`), le deuxième afin de détecter si le Khepera est à sa base (variable `Base`). Nous décrirons la patrouille sous la forme d'une séquence temporelle de commandes de haut niveau (le choix des comportements réactifs à appliquer) en prenant en compte certaines informations (ordre de patrouiller, niveau des réserves d'énergie, présence ou non de feu, …). Enfin, nous décrirons le programme final qui intégrera l'ensemble des briques de base définies préalablement.

4.1.4. Les variables utilisées

Sur la base des dix-huit variables sensorielles et motrices dont dispose le Khepera (`Px1`,…,`Px8`, `Lm1`,…, `Lm8`, `Md`, `Mg`) nous dérivons quatre variables sensorielles (`Dir`, `Prox`, `Theta` et `Base`) et deux variables motrices (`Vrot`, `Vtrans`) :

– **`Dir`** est une variable qui correspond approximativement à la direction de l'obstacle le plus proche (voir Figure 1). Elle prend des valeurs entre –10 (obstacle à gauche du robot) et +10 (obstacle à droite du robot) et est calculée de la manière suivante :

$$\texttt{Dir} = \text{Floor}\left(\frac{90(\texttt{Px6} - \texttt{Px1}) + 45(\texttt{Px5} - \texttt{Px2}) + 10(\texttt{Px4} - \texttt{Px3})}{9(1 + \texttt{Px1} + \texttt{Px2} + \texttt{Px3} + \texttt{Px4} + \texttt{Px5} + \texttt{Px6})}\right)$$

– **`Prox`** est une variable qui correspond à la proximité de l'obstacle le plus proche (voir Figure 1). Elle prend des valeurs comprises entre 0 (obstacle loin du robot) à 15 (obstacle très proche du robot) et est calculée de la manière suivante :

4. Phototaxie : Influence de la lumière sur le mouvement de petits organismes. Si le déplacement tend à se rapprocher de la source lumineuse on parle de phototaxie positive, dans le cas contraire, éloignement de la source lumineuse, phototaxie négative.

$$\texttt{Prox} = \text{Floor}\left(\frac{\text{Max}\left(\texttt{Px1}, \texttt{Px2}, \texttt{Px3}, \texttt{Px4}, \texttt{Px5}, \texttt{Px6}\right)}{64}\right)$$

– **Theta** est une variable qui indique la direction de la plus forte source lumineuse. Elle prend 36 valeurs comprises entre –170° et 180°. Cette variable sera définie par une description qui fusionnera les informations fournies par les capteurs de luminosité `Lm1` à `Lm8`.

– **Base** est une variable qui permet de détecter lorsque le Khepera est à sa base. Elle prend 2 valeurs (Vrai ou Faux). Cette variable sera définie par une description qui fusionnera les informations fournies par les capteurs de luminosité `Lm1` à `Lm8` et de proximité `Px1` à `Px8`.

– **Vrot** et **Vtrans** : le robot est piloté uniquement en vitesse de rotation (`Vrot`) (sa vitesse de translation, `Vtrans`, sera soit une valeur constante soit nulle). Il reçoit les commandes motrices par la variable `Vrot` calculée à partir de la différence entre la vitesse de sa roue gauche et la vitesse de sa roue droite. La variable `Vrot` peut prendre 21 valeurs comprises entre -10 (tourner à fond sur la gauche) et +10 (tourner à fond sur la droite).

$$\texttt{Mg} = \texttt{Vtrans} + \texttt{Vrot}$$
$$\texttt{Md} = \texttt{Vtrans} - \texttt{Vrot}$$

– **Feu** est une variable booléenne qui permet de détecter la présence d'un incendie. Au cours de sa patrouille, la détection de la présence éventuelle d'incendie sera obtenue à l'aide de la caméra linéaire du robot. Le Khepera utilisera sa micro turbine pour éteindre l'incendie en soufflant sur la bougie.

– **Pat** est une variable booléenne qui permet d'indiquer au robot s'il doit patrouiller ou s'arrêter et rentrer à sa base.

– **Eng** est une variable qui indique le niveau de charge des batteries du Khepera. Elle prend 4 valeurs {`th`, `h`, `b`, `tb`} pour très haut, haut, bas et très bas.

Quelques autres variables seront définies lors de la spécification des descriptions nécessaires à la réalisation de la tâche de surveillance.

4.1.5. Boucle de commande du robot

Pour contrôler notre robot en utilisant un programme bayésien nous appliquerons une décision tous les dixièmes de seconde. La question typique est de déterminer la valeur des variables motrices connaissant celles de variables sensorielles. Par conséquent, la boucle de commande pour contrôler le robot sera d'appliquer les instructions suivantes tous les dixièmes de seconde :

– Lire la valeur des capteurs ;

– $\texttt{motors} = \textbf{Draw}(\textbf{P}(\texttt{Motors} \mid \texttt{sensors} \otimes \omega))$;

– Appliquer les ordres moteurs ainsi obtenus.

4.2. Comportements réactifs

Notre premier objectif est de définir deux comportements élémentaires sous la forme de descriptions : la description ω–évit pour le comportement d'éviter les obstacles et la description ω–photo pour le comportement phototaxie. Nous définissons un comportement réactif comme une relation entre des variables sensorielles (dénotées par `Sensors`) et des variables motrices (dénotées par `Motors`), telle que la connaissance des valeurs sensorielles permet de déterminer les valeurs des variables motrices. Généralement, la description d'un comportement réactif sera exprimée avec la décomposition suivante :

$$\mathrm{P}\left(\texttt{Sensors} \otimes \texttt{Motors} \mid \omega\_react\right) = \begin{pmatrix} \mathrm{P}\left(\texttt{Sensors} \mid \omega\_react\right) \\ \times \mathrm{P}\left(\texttt{Motors} \mid \texttt{Sensors} \otimes \omega\_react\right) \end{pmatrix}$$

De manière générale, nous indiquons que dans le cadre de la description d'un comportement réactif, les différentes situations sensorielles sont équiprobables.

$$\mathrm{P}\left(\texttt{Sensors} \mid \omega\_react\right) \equiv \mathrm{Uniform}_{(\texttt{Sensors})}(\ ) = \frac{1}{\lfloor \texttt{Sensors} \rfloor}$$

En effet, nous considérons que la description d'un comportement réactif n'a pas pour objectif de décrire la répartition des conditions sensorielles. La description que nous spécifions s'intéresse uniquement à décrire la connaissance sur les variables motrices conditionnée par les valeurs des variables sensorielles. Ceci est obtenu en associant une forme paramétrique au deuxième terme de la décomposition :

$$\mathrm{P}\left(\texttt{Motors} \mid \texttt{Sensors} \otimes \omega\_react\right) \equiv \mathrm{F}_{\lambda_{\omega-react}(\texttt{Sensors})}\left(\texttt{Motors}\right)$$

À chaque situation sensorielle est associé un ensemble particulier de valeurs pour les paramètres λ. Ces paramètres peuvent être soit définis a priori par le programmeur, soit identifiés à partir d'un jeu de données expérimentales δ. L'utilisation typique de la description d'un comportement réactif consiste à déterminer les valeurs des variables motrices dans une situation sensorielle particulière, c'est-à-dire connaissant la valeur `sensors` des variables sensorielles. La réponse à cette question ne nécessite pas d'inférence, puisqu'en effet, elle correspond tout simplement au terme de la décomposition portant sur `Motors` : **P**(`Motors` | `sensors` ⊗ ω–react). Nous allons maintenant instancier ce schéma général pour réaliser les deux comportements réactifs, phototaxie et évitement d'obstacle.

4.2.1. Se diriger vers la lumière : description ω_photo

Description : *ω_photo*

Spécification :

Variables :

- $\texttt{Theta}$: $D_{\texttt{Theta}} = \{-170,\ldots,170,180\}$, $\lfloor \texttt{Theta} \rfloor = 36$
- $\texttt{Vrot}$: $D_{\texttt{Vrot}} = \{-10,-9,\ldots,10\}$, $\lfloor \texttt{Vrot} \rfloor = 21$

Décomposition :

$$P(\texttt{Theta} \otimes \texttt{Vrot} \mid \omega\_photo)$$
$$= P(\texttt{Theta} \mid \omega\_photo) \times P(\texttt{Vrot} \mid \texttt{Theta} \otimes \omega\_photo)$$

Formes :

$$P(\texttt{Theta} \mid \omega\_photo) \equiv U_{\lfloor \texttt{Theta} \rfloor}(\ )$$
$$P(\texttt{Vrot} \mid \texttt{Theta} \otimes \omega\_photo) \equiv G_{\mu(\texttt{Theta}),\sigma(\texttt{Theta})}(\texttt{Vrot})$$

Valeurs des paramètres :

$$\mu(\texttt{Theta}) = f\mu(\texttt{Theta})$$
$$\sigma(\texttt{Theta}) = f\sigma(\texttt{Theta})$$

Données expérimentales d' identification : $\varnothing$

Pour la spécification du comportement de phototaxie, la variable motrice est la vitesse de rotation du robot (`Vrot`) et la situation sensorielle est donnée par la direction de la source lumineuse (`Theta`). Nous pensons qu'une seule vitesse de rotation doit être préférée pour chaque situation sensorielle. Ainsi, la distribution conditionnelle **P**(`Vrot` | `Theta` ⊗ ω_photo) est représentée par une famille de lois uni modales de type gaussien[5]. Les valeurs des paramètres libres de la description sont définies par les fonctions suivantes :

$$f\mu(\texttt{Theta}) = 18 \times \left(sigmoid_{\alpha 1,\beta 1}(\texttt{Theta}) - 0.5\right) \quad \text{avec } \alpha 1 = 0 \text{ et } \beta 1 = 0,005$$
$$f\sigma(\texttt{Theta}) = 0,5 + 5 \times sigmoid_{\alpha 2,\beta 2}(|\texttt{Theta}|) \quad \text{avec } \alpha 2 = 90 \text{ et } \beta 2 = 0,012$$

5. ces formes sont des approximations discrètes d'une loi normale qui nécessite d'être normalisée sur le domaine de `Vrot`.

4.2.2. Éviter les obstacles : description ω_evit

Description : ω_evit

Spécification :

Variables :

- $\texttt{Dir}$: $D_{\texttt{Dir}} = \{-10,\ldots,10\}$, $\lfloor \texttt{Dir} \rfloor = 21$
- $\texttt{Prox}$: $D_{\texttt{Prox}} = \{0,\ldots,15\}$, $\lfloor \texttt{Prox} \rfloor = 16$
- $\texttt{Vrot}$: $D_{\texttt{Vrot}} = \{-10,\ldots,10\}$, $\lfloor \texttt{Vrot} \rfloor = 21$

Décomposition :

$$\mathrm{P}(\texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Vrot} \mid \omega\_evit)$$
$$= \mathrm{P}(\texttt{Dir} \otimes \texttt{Prox} \mid \omega\_evit) \times \mathrm{P}(\texttt{Vrot} \mid \texttt{Dir} \otimes \texttt{Prox} \otimes \omega\_evit)$$

Formes :

$$\mathrm{P}(\texttt{Dir} \otimes \texttt{Prox} \mid \omega\_evit) \equiv U_{\lfloor \texttt{Dir} \rfloor \times \lfloor \texttt{Prox} \rfloor}(\ )$$
$$\mathrm{P}(\texttt{Vrot} \mid \texttt{Dir} \otimes \texttt{Prox} \otimes \omega\_evit) \equiv G_{\mu(\texttt{Dir},\,\texttt{Prox}),\sigma(\texttt{Dir},\,\texttt{Prox})}(\texttt{Vrot})$$

Valeurs des paramètres :

$$\mu(\texttt{Dir},\ \texttt{Prox}) = t\_\mu[\texttt{Dir},\ \texttt{Prox}]$$
$$\sigma(\texttt{Dir},\ \texttt{Prox}) = t\_\sigma[\texttt{Dir},\ \texttt{Prox}]$$

Données expérimentales d' identification : δ_evit

La variable motrice retenue pour l'évitement d'obstacle est de nouveau la vitesse de rotation (`Vrot`). La situation sensorielle sera définie par la conjonction des variables `Dir` (direction de l'obstacle le plus proche) et `Prox` (proximité de l'obstacle le plus proche). Pour chaque situation sensorielle, nous pensons qu'il y a une seule vitesse de rotation préférentielle. Nous choisissons d'associer à **P**(`Vrot` | `Prox` ⊗ `Dir` ⊗ ω_evit) une distribution uni-modale. Cependant, selon la situation sensorielle, la décision devant être prise pour `Vrot` peut être plus ou moins certaine. Ceci est résumé en assignant une forme paramétrique de type loi normale à ce terme. À la différence du comportement phototaxie, cette fois-ci les paramètres libres de la description seront identifiés à partir de données expérimentales obtenues en téléopérant le Khepera avec un joystick. Tout d'abord, dans une phase d'apprentissage, nous pilotons le robot avec un joystick en lui faisant suivre un certain comportement, et collectons un ensemble de données δ_x associé à chaque comportement « x » particulier. Durant cette phase, le robot collecte tous les dixièmes de seconde les valeurs de ses variables sensorielles ainsi que les valeurs des commandes motrices (déterminé par la position du joystick). Appelons δ_evit l'ensemble particulier de données correspondant au comportement d'éviter les obstacles. Une donnée mesurée au temps t est un triplet ($\texttt{vrot}_t$, $\texttt{dir}_t$, $\texttt{prox}_t$). Les paramètres libres des formes paramétriques (moyennes et écart-types pour chacune des $\lfloor \texttt{Dir} \rfloor \times \lfloor \texttt{Prox} \rfloor$ gaussiennes) peuvent être mises à jour simplement au vu des

données expérimentales. Puis, dans une phase de restitution, le robot reproduit le comportement appris. Chaque dixième de seconde, il décide de la valeur à appliquer à ses variables motrices, connaissant la valeur de ses variables sensorielles et sa représentation interne de la tâche. Nous avons appliqué cette méthode pour apprendre par télé-opération différents comportements réflexes : éviter les obstacles (ensemble de données δ-evit), pousser les obstacles (δ-pousse), suivre le contour des obstacles (δ-suivre). Cette expérience est très souvent utilisée au cours de démonstrations grand public pour son caractère interactif et l'efficacité de ce type d'apprentissage pour les comportements réactifs. En effet, quelques dizaines de secondes d'apprentissage suffisent pour obtenir chacun des trois comportements par un manipulateur humain non expérimenté n'ayant aucune connaissance, ni du robot, ni du programme bayésien.

La principale leçon que nous retenons de cette expérience est qu'il est possible d'apprendre des comportements différents à partir des mêmes connaissances préalables en changeant uniquement les données servant à identifier les paramètres des distributions. Cette constatation montre clairement les rôles respectifs que jouent les connaissances préalables et les données expérimentales. Dans le cadre de cette expérience, les connaissances préalables expriment les choix et hypothèses retenus pour modéliser un ensemble de comportements réactifs sous la forme d'une dépendance entre la variable motrice `Vrot` et les informations fournies par les capteurs de proximités (résumées par les variables `Dir` et `Prox`). Ces connaissances décrivent comment un comportement réactif sera appréhendé du point de vue du robot, c'est-à-dire quelle structure interne au robot sera utile et nécessaire pour rendre compte d'une telle dépendance sensori-motrice. L'aspect fonctionnalité du comportement n'est donc pas spécifié au préalable, c'est par l'identification des paramètres libres de la description qu'une fonctionnalité particulière sera fixée.

4.3. Fusion sensorielle

L'objectif de cette section est de fournir un moyen de déterminer les deux variables `Theta` et `Base`. Ces variables seront obtenues en fusionnant des informations fournies par des capteurs de plus bas niveaux : les capteurs de luminosité (`Lm1` à `Lm8`) et les capteurs de proximité (`Px1` à `Px8`). La démarche que nous suivrons, consiste à considérer le problème de la perception (reconnaissance de situations, d'objets) comme un problème d'inversion[6]. Cette démarche peut se résumer de la manière suivante (Figure 2). Soit une situation de l'environnement pour laquelle nous choisissons un ensemble de paramètres afin de la caractériser et que nous résumons par la variable `V` (par exemple `Theta` les différentes orientations pour la source lumineuse, la variable booléenne `Base` : être ou ne pas être à la base). Le robot ne peut pas accéder directement à cette variable, c'est-à-dire qu'il ne dispose pas de capteur permettant de mesurer directement la valeur de cette

6. voir à ce propos (Poggio, 1984) dans le cadre de la vision.

variable. Par contre, il dispose de différentes informations sensorielles que nous notons S1, …, Sn.

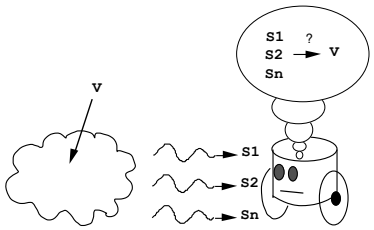

Figure 2. *La perception appréhendée comme un problème inverse.*

Ces informations sont influencées par le phénomène que l'on cherche à caractériser. Cette influence peut être modélisée dans le sens « causal » habituel, « connaissant le phénomène nous pouvons prédire la lecture d'une mesure sensorielle » : $\mathbf{P}(\mathtt{Si} \mid \mathtt{V})$

Si nous disposons de chacun de ces modèles élémentaires associés aux entrées sensorielles Si, en posant l'hypothèse d'indépendance conditionnelle que connaissant le phénomène physique (i.e. la variable V est connue), les entrées sensorielles sont supposées indépendantes[7]. :

$$\forall i \neq j$$
$$\mathrm{P}(\mathtt{Si} \otimes \mathtt{Sj} \mid \mathtt{V}) = \mathrm{P}(\mathtt{Si} \mid \mathtt{V}) \times \mathrm{P}(\mathtt{Sj} \mid \mathtt{V})$$

alors nous pouvons définir une description globale afin de fusionner les différents modèles élémentaires de la manière suivante :

$$\mathrm{P}(\mathtt{V} \otimes \mathtt{Si} \otimes \ldots \otimes \mathtt{Sn}) = \mathrm{P}(\mathtt{V}) \times \prod_{i=1}^{n} \mathrm{P}(\mathtt{Si} \mid \mathtt{V})$$

7. Attention, cette hypothèse exprime un choix de modélisation qui peut paraître étrange car de toutes évidences les différentes mesures sensorielles ne sont pas indépendantes. Nous exprimons ainsi l'idée que nous considérons le phénomène physique comme la principale raison de la contingence des lectures capteurs et que la variable V décrit « suffisamment » ce phénomène. Ainsi, nous posons que connaissant V, les lectures Si sont indépendantes, V est la cause des lectures et connaissant la cause, les conséquences sont indépendantes. C'est une hypothèse très forte. Les lectures peuvent être corrélées pour de nombreuses autres raisons qui ne sont pas prises en compte par V. Toutefois, dans une première approximation nous choisissons d'ignorer ces autres facteurs.

Déterminer la valeur de la variable V connaissant les valeurs si des variables sensorielles S1 à Sn revient à résoudre la question suivante :

$$P(V \mid s1 \otimes \ldots \otimes sn) = \frac{P(V) \times \prod_{1}^{n} P(si \mid V)}{P(s1 \otimes \ldots \otimes sn)} = \frac{1}{z} \times \prod_{1}^{n} P(si \mid V)$$

4.3.1. Déterminer l'orientation de la source lumineuse : description ω_theta

Description : *ω_lmi*

Spécification :

Variables :

- Lmi : $D_{Lmi} = \{0, \ldots, 511\}$, $\lfloor Lmi \rfloor = 512$
- Theta : $D_{Theta} = \{-170, \ldots, 170, 180\}$, $\lfloor Theta \rfloor = 36$

Décomposition :

$$P(Theta \otimes Lmi \mid \omega\_lmi) = P(Theta \mid \omega\_lmi) \times P(Lmi \mid Theta \otimes \omega\_lmi)$$

Formes :

$$P(Theta \mid \omega\_lmi) \equiv U_{\lfloor Theta \rfloor}(\ )$$

$$P(Lmi \mid Theta \otimes \omega\_lmi) \equiv G_{\mu(Theta),\sigma}(Lmi)$$

Valeurs des paramètres :

$$\mu(Theta) = f\mu(Theta, \theta_i)$$

$$\sigma = cste$$

Données expérimentales d' identification : $\varnothing$

La démarche que nous suivrons se décompose en deux étapes. La première étape consiste à spécifier, pour chaque capteur, une description (ω_lmi) qui définit la réponse du capteur connaissant la position angulaire de la source lumineuse par rapport au robot (ce qui correspond à définir **P**(Si | V) ci-dessus). Puis, dans la seconde étape, nous fusionnons les huit descriptions en une seule description (ω_theta), cette mise en commun pour définir **P**(V ⊗ S1 ⊗ … ⊗ Sn) afin de calculer **P**(V | S1 ⊗ … ⊗ Sn). La description du modèle de la réponse d'un capteur de luminosité porte sur deux variables : Lmi, la lecture du i^{ème} capteur de luminosité, et Theta la direction relative au robot de la source lumineuse. La décomposition spécifie simplement que la mesure de luminosité du capteur est conditionnée par la position de la source lumineuse. Nous ne privilégions pas certaines positions de la source lumineuse, nous fixons donc un a priori uniforme sur les valeurs de la variable Theta. Nous choisissons de représenter la probabilité de

Lmi par une famille de lois normales paramétrée par la valeur de Theta. L'écart type est fixé et est constant. La valeur moyenne est définie en fonction de Theta et du paramètre θ_i qui correspond à la position angulaire du $i^{ème}$ capteur sur le robot. Cette fonction est généralement facile à spécifier car elle correspond au type d'informations fournies par le fabriquant. Pour les capteurs de luminosité du Khepera, nous nous sommes inspirés des courbes données par le constructeur pour définir les moyennes des lois normales par la fonction suivante :

$$f\mu(\texttt{Theta},\ \theta_i) = 511 - \frac{1}{1+e^{-4\beta(|\texttt{Theta}-\theta_i|-\alpha)}},\ (\alpha = 45),\ (\beta = 0,03)$$

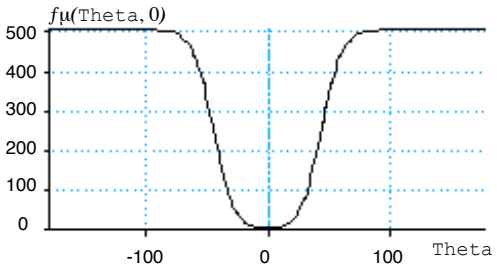

Figure 3. *fμ(Theta, 0), réponse d'un capteur de luminosité selon la position angulaire de la source lumineuse.*

Il n'y a pas de paramètres libres dans les connaissances préalables, il n'y a donc pas de phase d'identification. Toutefois, il pourrait être facile et intéressant de calibrer chacun des capteurs. Ceci peut être obtenu en identifiant les paramètres des lois normales par l'observation expérimentale de la réponse des capteurs pour une source lumineuse particulière. La fusion des huit modèles capteur ω_lm1 à ω_lm8 définie par la description ω_theta porte sur les neuf variables Theta, Lm1, …, Lm8. La décomposition de la distribution conjointe repose sur l'hypothèse d'indépendance suivante (conditionnellement à la connaissance de la valeur de la variable Theta, les différentes variables sensorielles Lmi sont indépendantes) :

$$\begin{aligned}&\mathrm{P}(\texttt{Theta} \otimes \texttt{Lm1} \otimes \texttt{Lm2} \otimes \ldots \otimes \texttt{Lm8} \mid \omega\_theta) \\ &= \mathrm{P}(\texttt{Theta} \mid \omega\_theta) \times \prod_{i=1}^{8} \mathrm{P}(\texttt{Lmi} \mid \texttt{Theta} \otimes \omega\_lmi)\end{aligned}$$

Cette décomposition est intéressante dans la mesure où elle permet d'exprimer d'une part un a priori sur l'orientation de la source lumineuse (ici nous avons fixé un

a priori uniforme) et, d'autre part, le produit des distributions décrivant la réponse `Lmi` de chacun des capteurs connaissant l'orientation de la source (ces distributions sont obtenues par une question posée à chacune des descriptions ω_lmi). Ce produit peut être vu comme une manière de fusionner les différentes informations provenant des capteurs de luminosité. Cette description est complètement définie, elle ne comprend pas de paramètres libres. Nous utilisons cette description en posant la question sur `Theta` connaissant les valeurs de luminosité :

$$\mathrm{P}\left(\texttt{Theta} \mid \texttt{lm1} \otimes \texttt{lm2} \otimes \ldots \otimes \texttt{lm8} \otimes \omega\_theta\right)$$
$$= \frac{1}{\Sigma} \times \prod_{i=1}^{8} \mathrm{P}\left(\texttt{lmi} \mid \texttt{Theta} \otimes \omega\_lmi\right)$$

La figure 4 montre les réponses des huit modèles élémentaires et le résultat de la fusion pour une source lumineuse située à 10° sur la droite du Khepera. L'information fournie par chacun des huit capteurs pris isolément est très médiocre, comme on peut l'observer sur les graphes en périphérie de la Figure 4. La fusion est extrêmement efficace (graphe au centre Figure 4) ; en rehaussant le signal elle permet, à partir de plusieurs capteurs de piètre qualité, d'obtenir une information très précise.

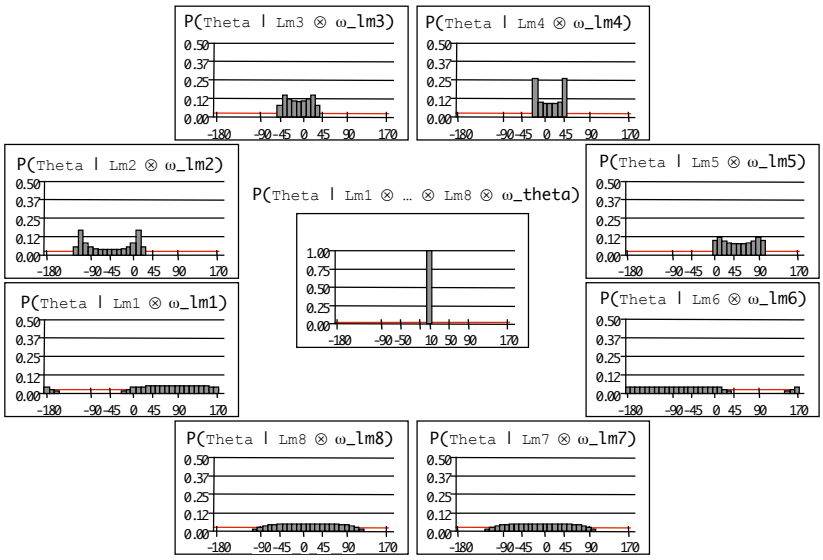

Figure 4. *Prédiction de `Theta` des huit modèles élémentaires et le résultat de la fusion pour une source lumineuse située à 10° sur la droite du Khepera.*

4.3.2. Reconnaître la base : description ω_base

L'objectif ici est de construire une description permettant au Khepera de reconnaître sa base lorsqu'il se trouve dans un endroit particulier de son environnement. Cette base est matérialisée dans un des coins de l'arène par un renfoncement fait de murs plus haut au dessus duquel est placée une petite lampe. Pour cette tâche de reconnaissance de situation, nous avons choisi de travailler avec les variables sensorielles de bas niveaux : capteurs de proximité (`Px1`, …, `Px8`) et de luminosité (`Lm1`, …, `Lm8`). Ce choix traduit l'hypothèse qu'il est possible de reconnaître la situation « être à la base » au vu simplement des valeurs retournées par ces variables. Cette situation sera résumée par la variable booléenne `Base`. Nous allons de nouveau appliquer un schéma de dépendance similaire à la description précédente ω_theta. Par l'expression d'hypothèses d'indépendance conditionnelle (conditionnellement à la connaissance de la valeur de la variable `Base`, les différentes variables sensorielles sont indépendantes), nous allons être en mesure de réduire considérablement la dimension pour représenter la distribution conjointe ($\lfloor \mathtt{Px} \rfloor^8 \times \lfloor \mathtt{Lm} \rfloor^8 \times \lfloor \mathtt{Base} \rfloor \approx 10^{46}$ cas).

Nous ne désirons pas spécifier de connaissance a priori sur la présence du Khepera à sa base. Nous associons donc une distribution uniforme à la probabilité d'être à la base (terme $\mathbf{P}(\mathtt{Base} \mid \omega\_base)$). Pour fixer les formes paramétriques des distributions élémentaires associées à chaque variable sensorielle, nous allons distinguer le cas où le robot se trouve à la base et le cas contraire. Lorsque le robot ne se trouve pas à la base, nous considérons qu'il peut se trouver dans une situation quelconque. Nous traduisons « situation quelconque » simplement en spécifiant que les capteurs sensoriels peuvent prendre une valeur quelconque (distribution uniforme). Dans le cas contraire (Khepera à sa base) la réponse des capteurs sensoriels sera représentée par une loi uni modale de type gaussien. L'identification des paramètres des gaussiennes (moyenne et écart-type) est très simple. Nous situons le Khepera à sa base et enregistrons pendant 30 secondes environ (soit 300 données échantillonnées à une cadence de 10Hz). Au cours de l'enregistrement, nous forçons le Khepera à effectuer quelques mouvements sur place de manière à éviter d'identifier une situation trop spécifique.

4.4. Combiner des comportements

Description : ω_ep
descriptions utilisées : ω_evit, ω_photo
Spécification :
Variables :
• Dir, Prox, Theta, H, Vrot
Décomposition :

$$P(\texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Theta} \otimes \texttt{H} \otimes \texttt{Vrot} \mid \omega\_ep) = \begin{vmatrix} P(\texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Theta} \mid \omega\_ep) \\ \times P(\texttt{H} \mid \texttt{Prox} \otimes \omega\_ep) \\ \times P(\texttt{Vrot} \mid \texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Theta} \otimes \texttt{H} \otimes \omega\_ep) \end{vmatrix}$$

Formes :

$$P(\texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Theta} \mid \omega\_ep) \equiv U_{\lfloor \texttt{Dir} \rfloor \times \lfloor \texttt{Prox} \rfloor \times \lfloor \texttt{Theta} \rfloor}(\ )$$

$$P(\texttt{H} \mid \texttt{Prox} \otimes \omega\_ep) \equiv f_{\texttt{Prox}}(\texttt{H})$$

$$P(\texttt{Vrot} \mid \texttt{Dir} \otimes \texttt{Prox} \otimes \texttt{Theta} \otimes \texttt{H} \otimes \omega\_ep) \equiv \begin{cases} \text{selon H} \\ [\texttt{H} = \texttt{evit}] : P(\texttt{Vrot} \mid \texttt{Dir} \otimes \texttt{Prox} \otimes \omega\_evit) \\ [\texttt{H} = \texttt{photo}] : P(\texttt{Vrot} \mid \texttt{Theta} \otimes \omega\_photo) \end{cases}$$

Valeurs des paramètres : pas de paramètres libres
Données expérimentales d' identification : ∅

L'objectif de cette expérience est d'illustrer une solution pour combiner les deux comportements, éviter les obstacles (description ω_evit) et se diriger vers la lumière (description ω_photo), afin d'obtenir un nouveau comportement qui permettra au Khepera de retourner à sa base (surmontée d'une source lumineuse) tout en évitant les obstacles. Les variables qui entrent en jeu pour la description de la combinaison sont naturellement les quatre variables associées aux deux descriptions des comportements de base, `Vrot`, `Dir`, `Prox`, `Theta`. Nous utiliserons une nouvelle variable `H` qui indique le type de comportement à suivre :

$$\texttt{H} : D_{\texttt{H}} = \{\texttt{evit}, \texttt{photo}\},\ \lfloor \texttt{H} \rfloor = 2$$

Nous décomposons la distribution conjointe en un produit de trois termes. Le premier terme porte sur les variables sensorielles `Dir`, `Prox`, `Theta`. Le deuxième exprime la dépendance de la variable `H` avec la situation sensorielle. Loin des obstacles, nous voulons que le robot se dirige vers la lumière et, inversement, proche des obstacles, le robot doit les éviter. Ainsi, nous considérons que la variable `H` ne doit dépendre que de la valeur de `Prox`. Finalement, le troisième terme exprime l'idée que la variable `Vrot` dépend des quatre autres variables.

La variable `H` permet de basculer entre l'évitement d'obstacle et la phototaxie. Lorsque la variable `H` à pour valeur `evit` le comportement appliqué est l'évitement d'obstacle tel qu'il est défini dans la description **P**(`Vrot` ⊗ `Dir` ⊗ `Prox` | ω_evit).

Lorsque H à pour valeur photo, le comportement appliqué est phototaxie décrit par **P**(Vrot ⊗ Theta | ω_photo). Enfin nous souhaitons spécifier une transition progressive entre les deux comportements en fonction de la valeur de la variable Prox. Cette transition est définie par la fonction suivante :

$$f_{\texttt{Prox}}(\texttt{H}) = \begin{cases} \text{selon H} \\ [\texttt{H} = \texttt{evit}] : \text{sigmoid}_{\alpha,\beta}(\texttt{Prox}) \qquad (\alpha = 9),(\beta = 0.25) \\ [\texttt{H} = \texttt{photo}] : 1 - \text{sigmoid}_{\alpha,\beta}(\texttt{Prox}) \end{cases}$$

La figure 5 illustre présente la probabilité de [H = evit] selon les valeurs de Prox.

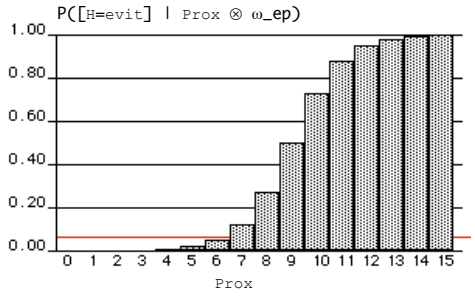

Figure 5. *Évolution de la probabilité de suivre le comportement d'évitement d'obstacle selon la valeur de la variable* Prox.

Il n'y a pas de paramètres libres dans les connaissances préalables, il n'y a donc pas de phase d'identification. La description est complètement définie et peut être utilisée. Nous utilisons cette description afin de déterminer la vitesse de rotation connaissant les valeurs des variables sensorielles. Par conséquent, afin de restituer cette combinaison de comportement nous posons la question suivante où la valeur de la variable H est ignorée :

$$\begin{aligned} & \text{P}(\texttt{Vrot} \mid \texttt{theta} \otimes \texttt{dir} \otimes \texttt{prox} \otimes \omega\_ep) \\ &= \sum_{\texttt{H}} \text{P}(\texttt{Vrot} \otimes \texttt{H} \mid \texttt{theta} \otimes \texttt{dir} \otimes \texttt{prox} \otimes \omega\_ep) \\ &= \begin{bmatrix} \text{P}(\texttt{evit} \mid \texttt{prox} \otimes \omega\_ep) \times \text{P}(\texttt{Vrot} \mid \texttt{dir} \otimes \texttt{prox} \otimes \omega\_evit) \\ + \text{P}(\texttt{photo} \mid \texttt{prox} \otimes \omega\_ep) \times \text{P}(\texttt{Vrot} \mid \texttt{theta} \otimes \omega\_photo) \end{bmatrix} \end{aligned}$$

Loin des obstacles (`Prox` = 0, **P**(`photo` | `Prox` ⊗ ω_ep) = 1) le robot applique un comportement purement phototaxie. Très proche des obstacles, (`Prox` = 15, **P**(`evit` | `Prox` ⊗ ω_ep) = 1) il effectue un pur évitement d'obstacle. Entre ces deux situations sensorielles extrêmes, la réponse à cette question montre que le robot effectue une combinaison pondérée des deux comportements (Figure 6). Cette situation (`Dir` = - 5, `Prox` = 8, `Theta` = - 90) correspond à un obstacle sur la gauche donc l'évitement favorise les vitesses de rotation positive (graphe en haut à droite) et une source lumineuse sur la gauche donc la phototaxie favorise les vitesses de rotation négative (graphe en haut à gauche)

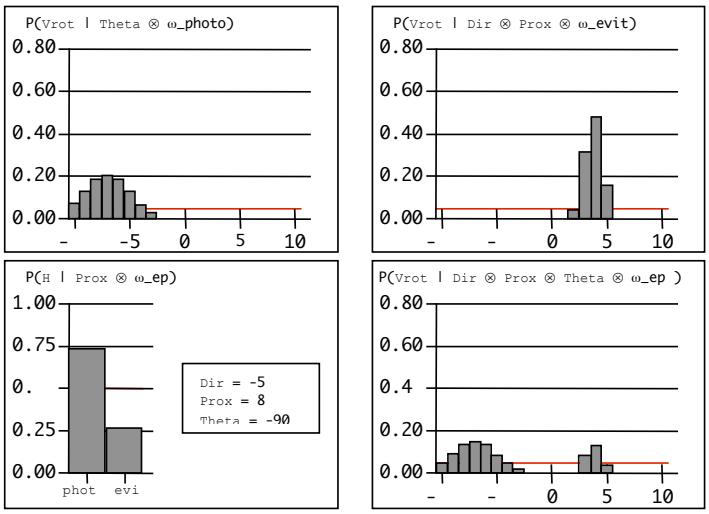

Figure 6. *Combinaison pondérée de l'évitement d'obstacle et de la phototaxie.*

Dans cette expérience, nous présentons une méthode générale pour combiner des descriptions, afin d'obtenir de nouveaux comportements. La méthode utilise une variable de décision `H` pour passer d'un des comportements combinés à l'autre. Une distribution de probabilités sur `H` sachant des variables sensorielles doit être soit spécifiée a priori, soit apprise expérimentalement (voir Diard et al., 1999, 2000). La description obtenue est finalement utilisée en posant des questions où `H` est inconnue. Le résultat de l'inférence est une somme sur les différents cas de `H`, ce qui correspond à un mélange pondéré des comportements combinés. Ceci montre que la PBR permet des spécifications de combinaisons simples, claires, et rigoureuses. Cela semble être un avantage important sur certaines autres approches, qui ont beaucoup de difficultés pour combiner ensemble des comportements, comme l'architecture de subsomption de Brooks (Brooks, 1986; Maes, 1989), ou les réseaux neuronaux. Cette méthode de combinaison semble également implémenter

naturellement un mécanisme similaire aux mixtures d'experts hiérarchiques (HEM, (Jordan et al., 1994))

4.5. Séquencer les comportements

Pour réaliser la tâche de surveillance, le robot va devoir effectuer des séquences temporelles de comportements réactifs. L'objectif de cette section est de montrer comment de telles séquences temporelles peuvent être spécifiées dans le cadre probabiliste. Au cours de la tâche de surveillance, le Khepera va effectuer successivement cinq types de tâches :

– *Inactif* : Le Khepera est à sa base, il ne bouge pas et attend pour repartir en patrouille qu'on lui en donne l'ordre (`Pat = true`), et que ses réserves énergétiques (`Eng`) soient suffisantes.

– *Patrouille* : Le Khepera se déplace aléatoirement dans l'environnement en évitant les obstacles.

– *Pompier* : Lorsqu'il détecte un incendie (`Feu = true`), le Khepera donne l'alarme et tente d'éteindre le feu. Les feux sont matérialisés par des bougies placées dans l'environnement du robot. Pour traiter les incendies, le Khepera dispose d'une micro-turbine qu'il peut mettre en action pour « souffler les bougies » (pgrm_pompier).

– *Retour* : En fin de patrouille (`Pat = false`) le Khepera retourne à sa base en suivant un comportement de phototaxie tout en évitant les obstacles. La base est une petite niche fortement éclairée par une lampe.

– *Energie* : Lorsque ses réserves d'énergie deviennent faibles, le Khepera retourne à sa base afin de recharger ses batteries.

Ces différents états sont représentés par la variable B_t, variable à 5 états {`inactif`, `patrouille`, `pompier`, `retour`, `energie`} Cette variable est mise en relation avec les variables `Pat`, `Base`, `Feu`, `Eng` (décrites section 4.1.4) et la variable B_{t-1} qui indique le type de tâche effectuée au pas de temps précédent. À chaque pas de temps, le robot va devoir décider quelle tâche accomplir sachant la tâche que l'on poursuivait à l'instant précédent (B_{t-1}) et sachant les valeurs des variables décrivant la situation courante (`Pat`, `Base`, `Feu` et `Eng`). La question qui va être posée sera donc :

$$P\left(B_t \mid B_{t-1} \otimes \texttt{Pat} \otimes \texttt{Base} \otimes \texttt{Feu} \otimes \texttt{Eng}\right)$$

Il est tentant de chercher à spécifier cette distribution directement. Ce serait suivre la démarche de programmation habituelle qui consiste, étant donné l'état à l'instant *t-1* et la situation courante, à déterminer ce qui doit être fait à l'instant *t*. Cette démarche peut s'avérer complexe car il peut rapidement y avoir un très grand nombre de conditions possibles (ici 160=5×2×2×2×4, alors que chacune des cinq

variables n'a que très peu de cas possibles). Nous proposons d'adopter une démarche de programmation dite « inverse ». En effet, il est simple, étant donné la tâche, d'avoir une idée sur les valeurs possibles des variables `Pat`, `Base`, `Feu` et `Eng` qui peuvent y correspondre. Par exemple, si le Khepera est en `patrouille` c'est qu'il en a reçu l'ordre et donc la variable `Pat` a nécessairement comme valeur `true`. De plus, on peut raisonnablement considérer que ces quatre variables sont indépendantes les unes des autres conditionnellement à la connaissance de la tâche en train de s'accomplir. Une telle décomposition offre notamment l'avantage de réduire considérablement la dimension du problème dans la phase de spécification.

$$\left[\begin{array}{l}\text{Décomposition :}\\ \left|\begin{array}{l}P\left(B_{t-1} \mid \omega\_cpt\right) \times P\left(B_t \mid B_{t-1} \otimes \omega\_cpt\right)\\ \times P\left(Pat \mid B_t \otimes \omega\_cpt\right) \times P\left(Base \mid B_t \otimes \omega\_cpt\right)\\ \times P\left(Feu \mid B_t \otimes \omega\_cpt\right) \times P\left(Eng \mid B_t \otimes \omega\_cpt\right)\end{array}\right.\end{array}\right.$$

Hormis l'a priori uniforme fixé à la variable Bt-1, les différentes formes seront définies sous la forme de tables de probabilités spécifiées par le programmeur :

$$\left[\begin{array}{l}\text{Formes :}\\ P\left(B_{t-1} \mid \omega\_cpt\right) \equiv \textit{Uniform}\\ P\left(B_t \mid B_{t-1} \otimes \omega\_cpt\right) \equiv \text{Table\_Bt}\left[B_t, B_{t-1}\right]\\ P\left(Pat \mid B_t \otimes \omega\_cpt\right) \equiv \text{Table\_Pat}\left[B_t, Pat\right],\ P\left(Base \mid B_t \otimes \omega\_cpt\right) \equiv \text{Table\_Base}\left[B_t, Base\right]\\ P\left(Feu \mid B_t \otimes \omega\_cpt\right) \equiv \text{Table\_Feu}\left[B_t, Feu\right],\ P\left(Eng \mid B_t \otimes \omega\_cpt\right) \equiv \text{Table\_Eng}\left[B_t, Eng\right]\end{array}\right.$$

Dans ces tables, nous utilisons la marque x pour indiquer que nous répartissons la « masse » de probabilité non spécifiée de manière équiprobable sur les différents cas marqués, de telle manière que la somme de tous les cas ait pour valeur 1. Par exemple, dans la deuxième colonne du tableau ci-dessous, x vaut 0,02 (0,02 = (1 – 0,8) / 4). Dans la Table 1, deux éléments essentiels sont spécifiés dans Table_Bt : d'une part, la « stabilité » (continuer à faire la même chose) du type d'action engagé par le Khepera qui peut être modulée (probabilité de 0,9 ou 0,99) ; et, d'autre part, l'interdiction de transition directe entre certains modes (probabilité 0). La Table 2 présente les probabilités des différentes variables décrivant la situation connaissant le type d'action engagé. Nous en commentons quelques cas :

– *Table_Pat* : lorsque le Khepera est dans le mode `inactif` (colonne 1), le plus probable est qu'il ne soit pas en veille (probabilité 0,9 pour [`pat = false`]). Cependant, il est possible qu'il soit en veille mais `inactif` pour cause de chargement de batterie. En mode `patrouille` (colonne 2) nous considérons qu'il ne peut être qu'en veille. A contrario, s'il est en mode `retour` (colonne 4), c'est nécessairement qu'on lui en a donné l'ordre et donc il ne peut plus être en veille. Les deux autres cas, `pompier` (colonne 3) et `énergie` (colonne 5), peuvent se

produire qu'il soit en `veille` ou non, on ne peut donc rien prédire sur la valeur de `Pat` dans ces cas (probabilité 0,5).

$P(B_t \mid B_{t-1} \otimes \omega\_cpt)$ ≡ Table_Bt		B_{t-1}				
		ina.	pat.	pom.	ret.	ene.
B_t	ina.	0.9	x	0	x	x
	pat.	x	0.9	0.01	x	x
	pom.	x	x	0.99	x	x
	ret.	0	x	0	0.9	x
	ene.	0	x	0	x	0.9

Table 1. *Table_bt définissant* **P**(`Bt` | `Bt-1` ⊗ ω_cpt).

P(Pat | B_t ⊗ ω_cpt) ≡ Table_Pat		B_t				
		ina.	pat.	pom.	ret.	ene.
Pat	false	0.9	0	x	1	x
	true	0.1	1	x	0	x
P(Base | Bt ⊗ ω_cpt) ≡ Table_Base						
Base	false	0	0.99	x	0.99	0.99
	true	1	0.01	x	0.01	0.01
P(Feu | Bt ⊗ ω_cpt) ≡ Table_Feu						
Feu	false	1	1	0	1	1
	true	0	0	1	0	0
P(Eng | Bt ⊗ ω_cpt) ≡ Table_Eng						
Eng	tb	0.325	0	x	x	0.8
	b	0.325	0.1	x	x	0.2
	h	0.25	0.45	x	x	0
	th	0.1	0.45	x	x	0

Table 2. *Tables définissant les probabilités des différentes variables décrivant la situation connaissant le type d'action engagé.*

– *Table_Base* : si le Khepera est `inactif` c'est qu'il est à sa base. Dans les modes `patrouille`, `energie` et `retour`, le Khepera ne se trouve pas à sa base (`Base = false`) toutefois cette éventualité n'est pas complètement impossible (probabilité de 0,01). Effectivement, au cours de l'exécution de ces différents modes, il existe une éventualité (même faible) qu'il passe par la base (et plus

particulièrement dans les modes `retour` et `energie`). Lors du traitement d'un incendie, nous considérons qu'il peut tout aussi bien être à sa base que ne pas y être.

– *Table_Feu* : cette table est très simple. Elle nous permet d'indiquer que le Khepera ne peut en aucun cas être en mode différent de `pompier` en présence d'un feu (`Feu = true`).

– *Table_Eng* : dans les cas où le robot est en mode `patrouille` (colonne 2) nous indiquons qu'il ne peut être dans ce mode lorsque que son énergie est très faible, et qu'il est possible, avec une probabilité relativement faible (0,1), qu'il ait un niveau d'énergie faible. Lorsqu'il est en mode `energie` (colonne 5), c'est que nécessairement son énergie est faible ([Eng = `b`] ou [Eng = t`b`]) avec une probabilité supérieure pour le niveau t`b` ; en aucun cas il ne peut avoir un niveau d'énergie élevé ([Eng = tf] ou [Eng = tf]). Pour les deux modes, `pompier` (colonne 3) et `retour` (colonne 4) nous ne pouvons rien prédire sur cette variable. Nous spécifions ainsi, implicitement, qu'en cas d'incendie ou d'ordre de fin de patrouille, le niveau énergétique importe peu et que le Khepera doit faire son possible soit pour intervenir soit pour rentrer à la base.

Le Khepera peut ainsi construire une séquence temporelle de tâches. Cette séquence temporelle s'avère très satisfaisante pour l'observateur. En particulier, elle est stable, le Khepera n'apparaît pas comme une « girouette » qui change d'avis tout le temps. La séquence est cependant très souple dans le sens où le Khepera change de tâche sans délais quand il le faut. Cette expérience illustre une méthode entièrement nouvelle et originale de construction de séries temporelles de tâches. Cette méthode pourrait être qualifiée de programmation « inverse ». En effet, on ne spécifie pas, comme il est d'usage, les conditions requises à l'application d'une certaine tâche. On donne, au contraire, pour chaque tâche, les observations associées, en considérant ces observations comme indépendantes les unes des autres sachant la tâche. La combinatoire des cas à envisager s'en trouve considérablement réduite. À l'usage, cette approche, tout d'abord déroutante, s'avère très naturelle et facile à suivre. De plus, elle est très adaptée pour l'apprentissage. En effet, dans un cas d'apprentissage supervisé, le professeur indique la tâche à réaliser à chaque instant, le robot observe les valeurs des autres variables et peut ainsi facilement mettre à jour les distributions correspondantes. Nous avons appliqué avec succès ce schéma d'apprentissage en robotique (Diard, 1999) ainsi que pour la programmation d'un agent dans un monde virtuel (LeHy, 2002).

4.6. Intégration : une tâche de surveillance

L'objectif de cette dernière description est de réaliser la tâche de surveillance telle qu'elle est décrite section 4.1.3. Pour construire cette description, nous utiliserons les différentes ressources présentées précédemment : ω_evit pour patrouiller aléatoirement en évitant les obstacles, ω_ep pour retourner à la base en mixant l'évitement d'obstacle et la phototaxie, ω_base pour détecter lorsque le

Khepera est à sa base, ω_theta pour déterminer l'orientation de la source lumineuse et enfin ω_cpt pour séquencer les différents comportements. 26 variables sont nécessaires :

– Vingt-deux variables sensorielles dont le Khepera sera en mesure de connaître la valeur tout les dixièmes de seconde : Px1 à Px8 (notés par la suite Px), Lm1 à Lm8 (notés par la suite Lm), Dir, Prox, Eng, Pat, Feu et B_{t-1} ;

– Trois variables internes [8] : Base, Theta et B_t ;

– Une variable motrice, Vrot, La valeur obtenue pour Vrot est appliquée aux moteurs.

Le choix de la décomposition de la distribution conjointe est motivé par l'objectif de vouloir décider des valeurs motrices connaissant les variables sensorielles. Cette décomposition est le produit d'une distribution uniforme sur les variables sensorielles avec trois questions adressées aux descriptions définissant les variables internes (ω_base, ω_theta, ω_cpt) et une dernière distribution qui définit le mode de contrôle moteur selon la valeur de la variable B_t :

$$\left[\begin{array}{l}\text{Décomposition :}\\ \\ P(Px \otimes Lm \otimes Dir \otimes Prox \otimes B_{t-1} \otimes Eng \otimes Pat \otimes Feu \otimes Base \otimes Theta \otimes B_t \otimes Vrot \mid \omega\_pat)\\ = \left|\begin{array}{l} P(Px \otimes Lm \otimes Dir \otimes Prox \otimes B_{t-1} \otimes Eng \otimes Pat \otimes Feu \mid \omega\_pat)\\ \times P(Base \mid Px \otimes Lm \otimes \omega\_base) \times P(Theta \mid Lm \otimes \omega\_theta)\\ \times P(B_t \mid Base \otimes B_{t-1} \otimes Eng \otimes Pat \otimes Feu \otimes \omega\_cpt)\\ \times P(Vrot \mid B_t \otimes Theta \otimes Dir \otimes Prox \otimes \omega\_pat)\end{array}\right.\end{array}\right.$$

Seule la dernière distribution nécessite d'être spécifiée. Elle correspond à une structure de branchement conditionnel selon la valeur de la variable B_t :

$$P(Vrot \mid B_t \otimes Theta \otimes Dir \otimes Prox \otimes \omega\_pat)$$
$$\equiv \left\{\begin{array}{l}\text{selon } B_t\\ \quad [B_t = inactif] : P(Vrot \mid \omega\_arrêt)\\ \quad [B_t = pompier] : P(Vrot \mid \omega\_pompier)\\ \quad [B_t = patrouille] : P(Vrot \mid Dir \otimes Prox \otimes \omega\_evit)\\ \quad [B_t = retour] \vee [B_t = energie] : P(Vrot \mid Theta \otimes Dir \otimes Prox \otimes \omega\_ep)\end{array}\right.$$

Les descriptions ω_pompier et ω_arrêt ne sont pas présentées dans cet article. Elles représentent des distributions sur Vrot de type fonction Dirac pour la valeur 0

8. Il existe une quatrième variable interne, la variable H de la description ω_ep ; toutefois la question qui sera posée à cette description considère cette variable comme cachée c'est pourquoi elle n'apparaît pas explicitement à ce niveau.

dans le cas ω_arrêt et pour une valeur calculée par un programme particulier (Pgrm_Pompier) qui réalise la tâche d'extinction d'incendie. L'utilisation de la description consiste à poser la question suivante, nous notons en minuscule les variables dont les valeurs sont connues :

$$P\left(\texttt{Vrot} \mid \texttt{px} \otimes \texttt{lm} \otimes \texttt{dir} \otimes \texttt{prox} \otimes \texttt{b}_{t-1} \otimes \texttt{eng} \otimes \texttt{pat} \otimes \texttt{feu} \otimes \omega\_pat\right)$$

« Quel ordre faut-il envoyer aux moteurs connaissant les variables sensorielles et en ignorant les variables internes ? ». La réponse à cette question est obtenue classiquement en sommant sur les variables ignorées et mène au résultat suivant :

$$\begin{aligned} & P\left(\texttt{Vrot} \mid \texttt{px} \otimes \texttt{lm} \otimes \texttt{dir} \otimes \texttt{prox} \otimes \texttt{b}_{t-1} \otimes \texttt{eng} \otimes \texttt{pat} \otimes \texttt{feu} \otimes \omega\_pat\right) \\ & = \sum_{\substack{\texttt{Theta} \\ \texttt{B}_t}} \left(\begin{array}{l} \left(\sum_{\texttt{Base}} \left(\begin{array}{l} P\left(\texttt{B}_t \mid \texttt{Base} \otimes \texttt{b}_{t-1} \otimes \texttt{eng} \otimes \texttt{pat} \otimes \texttt{feu} \otimes \omega\_cpt\right) \\ \times P\left(\texttt{Base} \mid \texttt{px} \otimes \texttt{lm} \otimes \omega\_base\right) \end{array} \right) \right) \\ \times P\left(\texttt{Theta} \mid \texttt{lm} \otimes \omega\_theta\right) \\ \times P\left(\texttt{Vrot} \mid \texttt{B}_t \otimes \texttt{Theta} \otimes \texttt{dir} \otimes \texttt{prox} \otimes \omega\_pat\right) \end{array} \right) \end{aligned}$$

Cette expression reflète exactement la structure du raisonnement nécessaire pour résoudre le problème : la somme la plus interne recherche la tâche courante à réaliser en ignorant la valeur de la variable Base.

$$\begin{aligned} & \sum_{\texttt{Base}} \left(\begin{array}{l} P\left(\texttt{B}_t \mid \texttt{Base} \otimes \texttt{B}_{t-1} \otimes \texttt{Eng} \otimes \texttt{Pat} \otimes \texttt{Feu} \otimes \omega\_cpt\right) \\ \times P\left(\texttt{Base} \mid \texttt{Px} \otimes \texttt{Lm} \otimes \omega\_base\right) \end{array} \right) \\ & = P\left(\texttt{B}_t \mid \texttt{B}_{t-1} \otimes \texttt{Eng} \otimes \texttt{Pat} \otimes \texttt{Feu} \otimes \omega\_cpt\right) \end{aligned}$$

Si aucune décision n'est faite excepté pour la variable `Vrot`, l'incertitude est propagée par la somation sur les variables internes. Toutes les informations disponibles sont prises en compte avec leur degré de certitude. Toutefois le coût de ceci peut être prohibitif. Afin de gagner en efficacité, il est possible de décider des valeurs des variables intermédiaires de la manière suivante :

– `base` = **Draw P**(`Base` | `px` ⊗ `lm` ⊗ ω_base) afin de décider si le robot est à sa base ou non ;

– $\texttt{b}_t$ = **Draw P**(`Bt` | `base` ⊗ $\texttt{b}_{t-1}$ ⊗ `eng` ⊗ `pat` ⊗ `feu` ⊗ ω_cpt) afin de décider le type de comportement à appliquer ;

– Selon la valeur de $\texttt{b}_t$ obtenue :

- `pompier` ou `arrêt` :appliquer directement les consigne pour `Vrot` fournie par le programme correspondant ;

- `patrouille` : `vrot` = **Draw P**(`Vrot` | `dir` ⊗ `prox` ⊗ ω_evit), décider de Vrot en appliquant le comportement d'évitement d'obstacle ;

- `retour` ou `energie` :

- `tetha` = **Draw P**(`Theta` | `lm` ⊗ ω_theta) décider de l'orientation de la source lumineuse puis,

- `vrot` = **Draw P**(`Vrot` | `dir` ⊗ `prox` ⊗ `theta` ⊗ ω_ep) décider de `Vrot` en mixant l'évitement d'obstacle et le comportement phototaxie.

Le résultat obtenu est très satisfaisant : le Khepera a rempli sa tâche de « veilleur de nuit » à de nombreuses occasions et dans des environnements variés. Cette expérience a tourné 12 heures par jour, sans interruption, durant 3 jours consécutifs dans le cadre de l'exposition la « Science en Fête ». Cette expérience permet d'illustrer que l'intégration en une seule et même description de toutes les descriptions était possible et aisée.

5. Conclusion

Nous avons introduit un nouveau formalisme nommé PBR pour programmer les robots en tenant compte de l'incomplétude des modèles choisis par le programmeur . Notre approche utilise le paradigme d'inférence bayésienne, et repose donc sur une base mathématique claire. Une tâche robotique est définie par une question posée à une description qui représente, sous la forme d'une distribution conjointe, la connaissance fournie par le programmeur, modulée éventuellement par des données expérimentales. Nous avons illustré la démarche pour programmer différentes tâches et montré comment un programme complexe peut être obtenu en combinant des modules plus simples.

En nous appuyant sur notre conception d'une tâche robotique (description + question) nous sommes en mesure de distinguer deux points durs : la construction de la description et son utilisation au moyen de l'inférence. Disons que l'élaboration d'une description est pour nous proche d'un problème de modélisation alors que son utilisation par l'inférence revêt plutôt un caractère calculatoire.

La mise au point d'une « bonne description » reste pour nous l'élément essentiel de la problématique de la programmation bayésienne et, pour l'instant, reste l'apanage du programmeur. Nous avons vu que nous pouvions scinder l'élaboration d'une description en trois points distincts : le choix des variables, la structure de dépendance et l'identification des paramètres. Ces choix ne sont pas indépendants et sont liés *in fine* à la tâche à accomplir. Toutefois, cette catégorisation des étapes de la modélisation offre un angle d'attaque qui permet d'envisager une certaine automatisation et, par cela, un pas vers l'autonomie.

Le choix des variables est un problème difficile et incontournable. Dans notre jargon, nous nous accordons à nommer cette première difficulté « le problème des pré-traitements ». Un pré-traitement peut être vu comme le résultat de l'application de traitements spécifiques sur les variables de base afin de rendre l'information plus *compacte* et plus *pertinente*, c'est-à-dire mieux *adaptée* à la tâche que l'on souhaite réaliser. De plus, et c'est bien souvent le cas, à l'aide d'un pré-traitement nous cherchons à obtenir, à partir des signaux fournis par les capteurs, une information plus aisément *interprétable* du point de vue du programmeur. Enfin, nous pouvons souhaiter obtenir une ressource *générique* pour programmer, c'est-à-dire suffisamment riche pour pouvoir être utilisée dans d'autres circonstances ou à d'autres fins. Ces différents objectifs sont rarement compatibles entre eux et, en pratique, aboutissent aux choix et à la mise en œuvre de différentes techniques pour les atteindre. Ces techniques sont nombreuses (échantillonnage, filtrage, dérivation, intégration, transformation logarithmique, analyse de données par regroupement, « clustering », analyse en composantes principales, quantification vectorielle, …). Elles recouvrent un large éventail de travaux allant de la robotique aux techniques d'analyse de données et de traitement du signal. Des techniques de mesure d'information (comme la mesure de Kullback-Leibler ou Entropie relative) offrent un cadre intéressant pour quantifier l'adéquation relative d'un pré-traitement. Un pré–traitement peut être également obtenu en mettant en relation les informations fournies par les variables de base avec des modèles abstraits de l'environnement. Nous avons illustré ce point dans l'exemple du modèle capteur (section 4.3) en montrant que l'approche bayésienne traite ce problème comme un problème d'inversion.

Le choix de la structure de dépendance et des formes paramétriques associées est lui aussi délicat. Les données expérimentales permettent de fixer les valeurs des distributions de probabilités, mais elles peuvent également servir à déterminer la structure de dépendance et les formes paramétriques. Dans les réseaux bayésiens, il existe des algorithmes qui permettent d'identifier expérimentalement le graphe du réseau. En pratique, nous nous accordons la liberté de fixer à notre convenance les formes paramétriques ainsi que la valeur des paramètres. Toutefois, il est clair que si nous y gagnons ainsi une plus grande richesse d'expression, nous sortons de la rigueur offerte par l'application de principes généraux tels que celui du maximum d'Entropie. Dans ces cas, les solutions choisies deviennent plus difficilement justifiables voir simplement discutables.

L'inférence bayésienne, nous l'avons illustré, est en mesure de répondre à toute question portant sur une partie des variables. Généralement, le calcul de la solution nécessite une marginalisation sur les variables ignorées. Cette marginalisation consiste à déterminer (ou estimer) une distribution de probabilité puis à rechercher les solutions les plus probables. Pour les expériences présentées, les dimensions des espaces de marginalisation étaient suffisamment faibles pour qu'une énumération exhaustive soit envisagée. Toutefois, cette stratégie n'est plus tenable pour des espaces de recherche de grande dimension. Cette difficulté ne peut être abordée que

par le biais d'inférence approximée comme un problème d'optimisation ; en marginalisant, nous cherchons finalement les « pics » de probabilité dans un espace pouvant être très grand. Différentes techniques existent pour estimer la marginalisation et trouver les solutions les plus probables ; citons les algorithmes génétiques, les algorithmes de type « recuit simulé », les méthodes de Monte-Carlo. Dans certains cas, on peut envisager d´autres méthodes de résolutions comme (Ruiz, 1998) dans le cadre des mixtures de gaussiennes, (Jaakkola, 1999) dans le cadre des modèles graphiques. En fait, ces méthodes de résolution sont spécifiques et indissociables d'un ensemble d'hypothèses et de choix de modélisation du problème traité. Ainsi, nous sommes conduit à reconnaître, qu'à terme, seules les connaissances préalables permettent de rendre un problème traitable en pratique.

12. Bibliographie

Aji S. M., McEliece R. J., « The generalized distributive law », *IEEE Transaction of Information Theory*, vol. 46, n°2, 2000, p. 325-343.

Alami R., Chatila R., Fleury S., Ghallab M. & Ingrand F, « An architecture for autonomy », *International Journal of Robotics Research*, vol. 17, n° 4, 1998, p. 315-337.

Arulampalam, S., Maskell, S., Gordon, N. & Clapp, T., « A tutorial on particle filters for on–line non-linear/non-Gaussian bayesian tracking », *IEEE Transaction of Signal Processing*, vol. 50, n° 2, 2002, p.174-188.

Aycard O., Architecture de contrôle pour robot mobile en environnement intérieur structuré, Thèse de doctorat, Univeristé Henri Poincaré, Nancy, 1998.

Bernhardt, R. & Albright, S.L., éditeurs, *Robot calibration*, Londre, Chapman & Hall, 1993.

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K., « Interprétation ou description (I) : proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs », *Intellectica*, vol. 26-27, p. 257-311, Paris, 1998.

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K., « Interprétation ou description (II) : fondements mathématiques de l'approche F+D », *Intellectica*, vol. 26-27, p. 313-336, Paris, 1998.

Bessière, P. & al., Survey : Probabilistic methodology and techiques for artefact conception and development, rapport de recherche n° RR–4730, 2003, INRIA.

Borrelly, J-J., Coste, E., Espiau, B., Kapellos, K., Pissard-Gibollet, R., Simon, D. & Turro, N., « The ORCCAD architecture », *International Journal for Robotics Research*, vol. 17, n° 4, 1998, p. 338-359.

Brafman, R.I., Latombe, J-C., Moses, Y. & Shoham, Y., « Applications of a logic of knowledge to motion planning under uncertainty », *Journal of the ACM*, vol. 44, n° 5, 1997, p. 633-668.

Bretthorst, G.L., « Bayesian spectrum analysis and parameter estimation », *Lecture Notes in Statictics*, New York, Spinger-Verlag, vol. 48, 1988.

Brooks, R.A., « A robust layered control systems for a mobile robot », *IEEE Journal of Robotics and Automation*, vol. 2, n° 1, 1986, p. 14-23.

Cooper, G., « The computational complexity of probabilistic inference using Bayesian belief networks », *Artificial Intelligence*, vol. 42, 1990, p. 393-405.

Cox, R.T., *The algebra of probable inference*, The John Hopkins Univ. Press, Baltimore, 1961.

Dagum, P. & Luby, M., « Approximate probabilistic reasoning in Bayesian belief networks is NP-hard », *Artificial Intelligence*, vol. 60, 1993, p. 141-153.

Darwiche, A. & Provan, G., « Query DAGs : a practical paradigm for implementing belief–networks inference », *Journal of Artificial Intelligence Research*, vol. 6, 1997, p. 147-176.

Dekhil, M. & Henderson, T.C., « Instrumented sensor system architecture », *International Journal for Robotics Research*, vol. 17, n° 4, 1998, p. 402-417.

Delcher, A.L., Grove, A.J., Kasif, S. & Pearl, J., « Logarithmic-time updates and queries in probabilistic networks », *Journal of Artificial Intelligence Research*, vol. 4, 1996 , p. 37–59.

Diard, J. & Lebeltel, O., « Bayesian learning experiments with a Khepera robot », *Experiments with the Mini-Robot Khepera Proceedings of the 1st International Khepera Workshop*, 1999, Löffler Mondada Rückert éditeurs, Paderborn, HNI-Verlagsschriftenreihe, Band 64, Germany, p. 129-138.

Diard, J. & Lebeltel, O., « Bayesian programming and hierarchical learning in robotics », Meyer, Berthoz, Floreano, Roitblat & Wilson éditeurs, *SAB2000 Proceedings Supplement Book*, Publication of the International Society for Adaptive Behavior, Honolulu, 2000.

Donald, B.R., « A geometric approach to error detection and recovery for robot motion planning with uncertainty », *Artificial Intelligence*, vol.37, 1988, p. 223-271.

Erickson, G.J. & Smith, C.R., *Maximum-Entropy and Bayesian methods in science and engineering , vol. 1 : Foundations*, Kluwer Academic Publishers, 1988.

Erickson, G.J. & Smith, C.R., *Maximum-Entropy and Bayesian methods in science and engineering , vol. 2 : Applications*, Kluwer Academic Publishers, 1988.

Frey, B.J., *Graphical Models for Machine Learning and Digital Communication*, MIT Press, 1998.

Fox, D., Burgard, W., Kruppa, H. & Thrun, S., « A probabilistic approach to collaborative multi-robot localization », *Autonomous Robots*, vol. 8, 2000, p. 325-344.

Gutmann, J-S., Burgard, W., Fox, D. & Konolidge, K., « Experimental comparison of localization methods », *proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

Halpern J.Y., « A counterexample to theorems of Cox and Fine », *Journal of Artificial Intelligence Research*, vol. 10, 1999, p. 67-85.

Halpern, J.Y., « Cox's theorem revisited », *Journal of Artificial Intelligence Research*, vol. 11, 1999, p. 429-435.

Jaakkola, T.S., Jordan M.I., « Variational probabilistic inference and the QMR-DT network », *Journal of Artificial Intelligence Research*, vol. 10, 1999, p. 291-322.

Jaynes, E.T., « Where do we stand on maximum entropy? », *The maximum entropy formalism*, D. Levine & M. Tribus éditeurs, MIT Press, 1979 .

Jaynes, E.T., « On the rationale of maximum-entropy methods » *Proceedings of the IEEE*, vol. 70, 1982, p. 939-952.

Jaynes, E.T., *Probability theory - The logic of science*, Cambridge University Press, 2003.

Jensen, F., Lauritzen S., Olesen K., « Bayesian updating in recursive graphical models by local compuations », *Computational Statistic Quaterly*, vol. 4, 1990, p. 269-282.

Jordan, M.I. & Jacobs R.A. « Hierarchical mixtures of experts and the EM algorithm », *Neural Computation*, vol. 6, 1994, p. 181-214.

Jordan, M.I., *Learning in Graphical Models*, Cambridge MA , MIT Press, 1998.

Jordan, M.I., Ghahramani, Z., Jaakkola, T.S. & Saul, L., « An introduction to variational methods for graphical models », *Machine Learning*, vol. 37, n° 2, 1999, p. 183-233.

Kaelbling, L.P., Littman, M.L. & Cassandra, A.R., « Partially observable Markov decision processes for artificial intelligence », *Proceedings of International Workshop Reasoning with Uncertainty in Robotics*, Springer-Verlag, 1996, p.146-62.

Kaebling, L.P., Cassandra, A.R. & Kurien, J.A., « Acting under uncertainty : discrete bayesian models for mobile-robot navigation », *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

Kaebling, L.P., Littman, M.L. & Cassandra, A.R., « Planning and acting in partially observable stochastic domains », *Artificial Intelligence*, vol. 101, 1998.

Kapur, J.N., & Kesavan, H.K., *Entropy optimization principles with applications*, Academic Press, Inc., 1992.

Koller, D., & Pfeffer, A., « Object-oriented bayesian networks », *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997, p. 302-313.

Konolidge, K., « Improved occupancy grids for map building », *Autonomous Robots*, vol. 4, 1997, p. 351-367.

Konolidge, K. & Chou, K., « Markov localization using correlation », *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, Washington, 1999, p. 1154-1159.

Lane, T. & Kaebling, L.P., « Toward hierachical decomposition for planning in uncertain environments », *Workshop on planning under Uncertainty and Incomplete Information at the 2001 International Joint Conference on Artificial Intelligence*, 2001.

Laplace, Pierre Simon de, « Mémoire sur la probabilités des causes par les évènements », Mémoire de l'académie royale des sciences; réimprimé dans *Oeuvres complètes de Laplace*, vol. 8, Gauthier Villars, Paris, France, 1774.

Laplace, Pierre Simon de, « Essai philosophique sur les probabilités », Courcier Imprimeur, Paris; réimprimé dans *Oeuvres complètes de Laplace*, vol. 7, Gauthier Villars, Paris, France, 1814.

Lauritzen, S. & Spiegelhalter, D., « Local computations with probabilities on graphical structures and their application to expert systems », *Journal of the Royal Stastical Society*, vol. 50, n°19, 1988, p. 157-224.

Lauritzen, S. L., *Graphical Models*, Oxford Univ. Press, Oxford, 1996.

Lebeltel, O., Programmation Bayésienne des Robots, Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, 1999.

Lozano-Perez, T., Mason, M.T., Taylor, R.H., « Automatic synthesis of fine-motion strategies for robots », *International Journal of Robotics Research*, vol. 3, n° 1, 1984, p. 3-24.

MacKay, D. G., « Introduction to Monte-Carlo methods », *Learning in Graphical Models*, M. Jordan M. ed., MIT Press, 1996, p. 175-204..

Maes, P., « How to Do the Right Thing », *Connection Science Journal*, vol. 1, n°3, 1989, pp. 291-323.

Matalon, B., « Epistémologie des probabilités », *in* Jean Piaget (sous la direction de), *Logique et connaissance scientifique*, Paris, Gallimard, « La Pléiade », 1967, p. 927-991.

Mazer, E., Boismain, G., Bonnet des Tuves, J., Douillard, Y., Geoffroy, S., Dubourdieu, J., Tounsi, M. & Verdot, F., « START: an industrial system for teleoperation », *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1998, p. 1154-1159.

Mekhnacha, K., Mazer, E. & Bessière, P., « The design and implementation of a bayesian CAD modeler for robotic applications », *Advanced Robotics*, vol. 15, n° 1, 2001.

Mohammad-Djafari, A.& Demoment, G., *Maximum entropy and bayesian methods*, Kluwer Academic Publishers, Paris, France, 1992.

Neal, R.M., Probabilistic inference using Markov chain Monte-Carlo Methods, Technical Report CRG-TR-93-1, Departement of computer Science, University of Toronto, 1993.

Pogio, T., « Low-level Vision as Inverse Optics », *in Proceedings of Symposium on Computational Models of Hearing and Vision*, M. Rauk ed ., Academy of Science of the Estoninan S.S.R., 1984, p. 123-127.

Robert, C., « An entropy concentration theorem: applications », *in artificial intelligence and descriptive statistics, Journal of Applied Probabilities*, 1990.

Rosenblatt, J.K., « Optimal selection of uncertain actions by maximizing expected utility » ; *Autonomous Robots*, vol. 9, n° 1, 2000, p. 17-25.

Ruiz, A., Lopez-de-Teruel, P.E. & Garrido, M.C., « Probabilistic inference from arbitrary uncertainty using mixtures of factorized generalized gaussians », *Journal of Artificial Intelligence Research*, vol. 9, 1998, p. 167-217.

Saul, L.K., Jaakkola, T. and Jordan, M.I., « Mean field theory for sigmoid belief networks », *Journal of Artificial Intelligence Research*, vol. 4, 1996, p. 61-76.

Schneider, S.A., Chen, V.W., Pardo-Castellote, G., Wang, H.H., « ControlShell : a software architecture for complex electromechanical systems », *International Journal for Robotics Research*, vol. 17, n° 4, 1998, p. 360-380.

Smith, C.R. & Grandy, W.T. Jr., *Maximum-Entropy and bayesian methods in inverse problems*, D. Reidel Publishing Company, Holland, 1985.

Tarentola, A., *Inverse problem theory: methods for data fitting and model parameters estimation*, Elsevier, New York, USA, 1987.

Thrun, S., « Bayesian landmark learning for mobile robot localization », *Machine Learning*, vol. 33, n° 1, 1998, p. 41-76.

Thrun, S., Burgard, W., Fox, D., « A probabilistic approach to concurrent mapping and localization for mobile robots », *Autonomous Robots*, vol. 5, 1998, p. 253-271.

Thrun, S., Probabilistic algorithms in robotic, Tech. Report CMU-CS-00-126, Computer Science Departement, Carmegie Mellon University, 2000.

Zhang, N.L. and Poole, D., « Exploiting causal independence in bayesian network inference », *Journal of Artificial Intelligence Research*, vol. 5, 1996, p. 301-328.